Paper SAS4493-2020

Neural Network–Based Forecasting Strategies in SAS® Viya®

Steven C. Mills, SAS Institute Inc.

ABSTRACT

Recent literature indicates that hybrids of machine learning and classical time series models are among the top contenders in accurately forecasting the future. Classical linear models are parsimonious and often perform well, but they are unable to capture nonlinear relationships in the data. On the other hand, machine learning models such as neural networks (NNs) are very good at modeling nonlinear effects. Knowing when and how to use machine learning models might seem difficult, but these decisions can be distilled down to best practices that any analyst can use with little experience. This paper discusses several NN-based modeling strategies available in SAS® Visual Forecasting software and the important factors to consider in choosing and training a model. The discussion includes key features of the data that inform the decision to use machine learning models, feature generation options to augment the training process, and best practices to fit a robust model. This knowledge will enable you to leverage the advantages of both NN and linear models to achieve more powerful forecasts.

INTRODUCTION

Machine learning and hybrid modeling strategies have emerged as top contenders in time series forecasting because of the volume of data and processing power brought about by the information age. Neural networks have become particularly popular because they are able to approximate any functional relationship and they are very well suited for modeling nonlinear relationships between the dependent (target) variable and independent (predictor) variables (Box 1976, Yoshio, Hipel and Mcleod 2005, Taşpınar 2015, Crone and Häger 2016).

SAS Visual Forecasting implements three forecasting strategies that are based on neural networks (NNs): panel series neural network, stacked model, and multistage model. Neural networks might seem mysterious or even intimidating because they have many parameters, but the guidelines in this paper will enable you to successfully apply NNs to forecasting problems and increase your forecasting accuracy. The first section explains how each of the three NN-based modeling strategies is customized for time series forecasting and what types of data work well. Next, a case study shows predictions of ozone levels in Chicago by using an NN-based strategy that easily outperforms classical models. Finally, some effective use strategies not covered in the example are discussed.

After reading this paper, you will understand what types of data work well with these modeling strategies and you will be able to effectively apply these strategies to your own forecasting problems. Whether the volume of data is medium size, big, or huge, these modeling strategies can help identify complex relationships between variables and increase the predictive power of your models.

BACKGROUND

Some basic knowledge of neural networks is presented here in order to provide a foundation for the concepts discussed later. A neural network is composed of an input layer, one or more hidden layers, and an output layer. An example NN with one hidden layer is shown in Figure 1a. Each input node has a connection to every node in the first hidden layer. Likewise, every node in the hidden layer has a connection to the node in the output layer.

Figure 1b expands the hidden node n_4 to illustrate how the output of a node is calculated. (Nodes n_3 and n_5 are omitted for clarity.) The input layer simply passes through the values in the input vector X such that the output of node n_1 is the value x_1 and the output of n_2 is the value x_2. The output of each node is multiplied by a connection, w_{jk}, where *j* and *k* represent the nodes being connected. The hidden node adds up the input values and a bias parameter, b_k, and feeds the sum into a nonlinear activation function to produce the hidden node output. The output from n_4 is multiplied by the corresponding connection weight, w_{46}, and fed forward to node n_6.

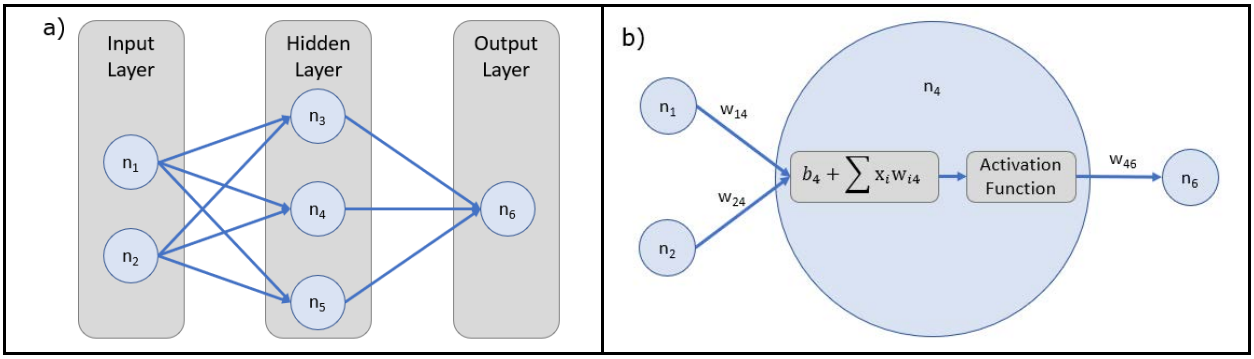

Figure 1. (a) Example Neural Network Architecture and (b) Functionality of a Node

Neural networks learn through an iterative cycle of training and validation. Training data are fed forward through the NN to calculate an output value. Then back propagation is used to update the connection weights and reduce the error. The details of back propagation are outside the scope of this paper, but they can be summarized succinctly as follows: The error is measured with respect to the training data, and partial derivatives are calculated with respect to each connection weight. The aggregated partial derivatives are used to update the connection weights and reduce the error (Goodfellow, Yoshua and Courville 2016). The error with respect to the validation data is measured periodically to validate the model training process.

NEURAL NETWORK MODELING IN SAS VISUAL FORECASTING

Some special considerations are required to successfully use NN-based models in time series forecasting. Fortunately, the modeling strategies in SAS Visual Forecasting take care of a lot of the details by structuring the data and generating additional features before training the model. However, it is important to understand how the data are interpreted and when to use the extracted features. This section describes how the NN-based strategies in SAS Visual Forecasting structure interpret the data compared to classical forecasting and machine learning.

DATA STRUCTURE

In classical time series forecasting, BY variables are used to delineate a panel of related time series and find the best model for each series individually. This approach is intractable for NNs because they require significantly more data to train than classical models require. A single time series is typically not enough.

The neural network modeling strategies in SAS Visual Forecasting are designed for panel data that consist of multiple related time series. For example, a retail chain might have many time series that are delineated by BY variables (such as STORE and SKU) and independent variables (such as promotions, number of shoppers, and calendar events).

The time series are concatenated together as shown in Figure 2 and modeled as a single series with the BY variables included as categorical independent variables. The resulting input table is a concatenation of all the series.

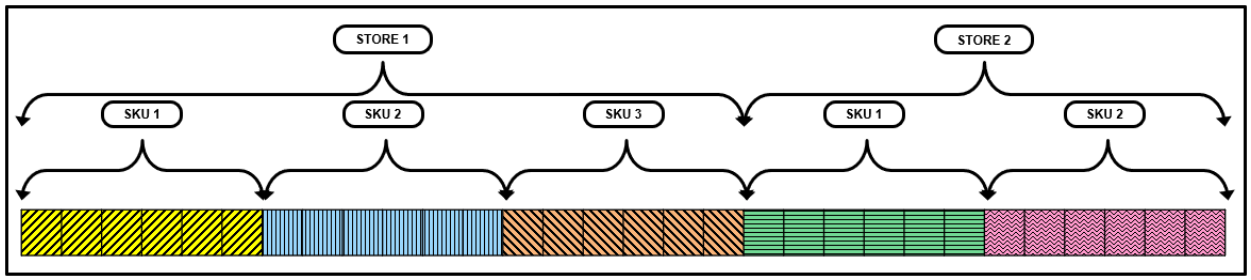

Figure 2. Five Concatenated Time Series Delineated by the BY Variables STORE and SKU

DATA PARTITIONING

A typical machine learning algorithm randomly partitions the data into training, validation, and test partitions. Model fitting is accomplished through alternating cycles of using the training data to minimize the training partition error and validating the progress by checking the validation partition error. This cycle continues until some stopping criterion is met.

The training and validation partitions are analogous to the training and holdout samples in time series forecasting where the holdout portion is used to select the model that best generalizes to new data. The out-of-sample (test) data are used to measure how well the model predictions generalize to new data.

Random sampling in order to partition data is acceptable for machine learning applications, but in time series the most recent data usually have a larger impact on predicted values. To adjust for this, the NN-based forecasting strategies use ordered sampling. Random and ordered sampling are illustrated in Figure 3 for a single time series. In ordered sampling, the oldest data are placed in the training partition and more recent data are placed in the holdout (validation) partition. The out-of-sample (test) partition contains the most recent data leading up to the forecast horizon.

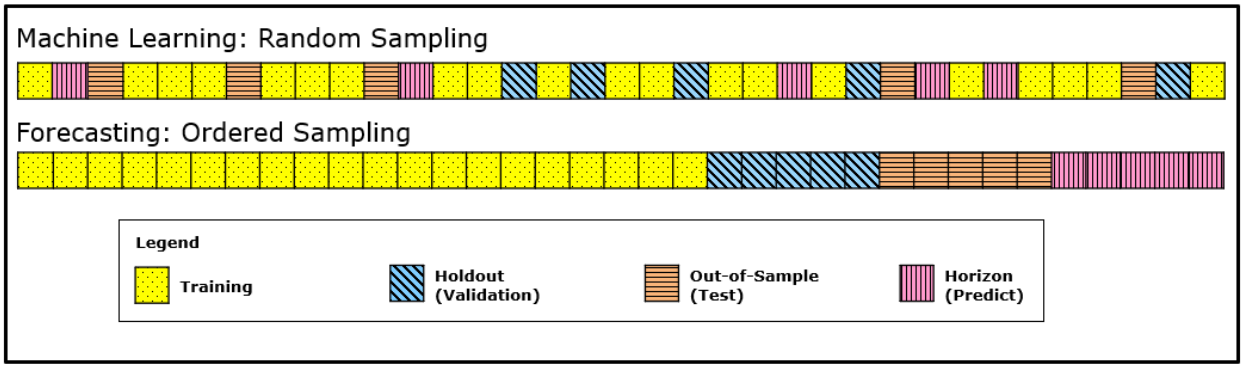

Figure 3. Partitioning for Machine Learning (Random Sampling) vs. Forecasting (Ordered Sampling)

MODELING STRATEGIES

Panel Series Neural Network

The panel series neural network (PSNN) can be used to implement a neural network like the one described in the introduction. Figure 4 illustrates the general flow of operations in the PSNN modeling strategy. The input data first go through a preprocessing step where data are partitioned, transformed, and/or standardized. Next, salient features are extracted. After preprocessing and feature extraction, the NN learns how to fit a model to the data. Finally, the output data are reverse-transformed and destandardized back to the original scale to produce the final forecast.

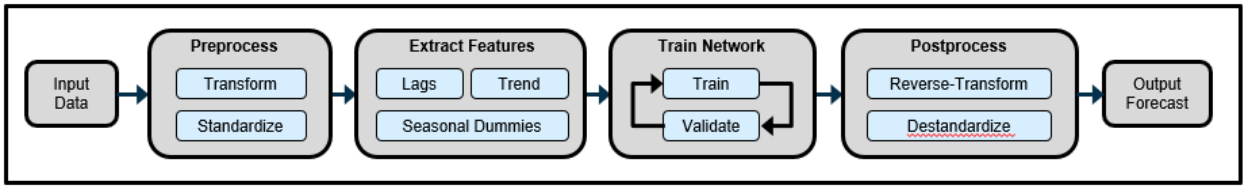

Figure 4. Panel Series Neural Network Modeling Strategy

Stacked Model

The stacked model uses the PSNN to create an initial forecast of the target variable and then models the residuals by using a classical time series approach. The forecasts of the input data and the residuals are then added together to generate the final forecast, as shown in Figure 5.

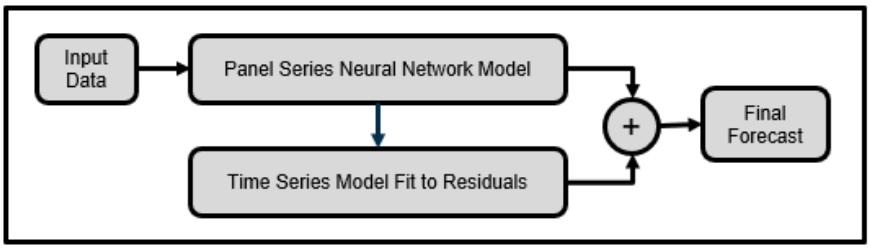

Figure 5. Stacked Modeling Strategy

Multistage Model

The multistage model is a twist on hierarchical forecasting. The lowest levels in a time series hierarchy are often intermittent or display more nonlinear characteristics that challenge time series models. Figure 6a shows the block diagram, and Figure 6b shows an example hierarchy. The lower levels are modeled by using regression or a neural network, whereas the higher levels are fit to time series models. The forecasts are reconciled at a user-specified level of the hierarchy to produce the final forecast.

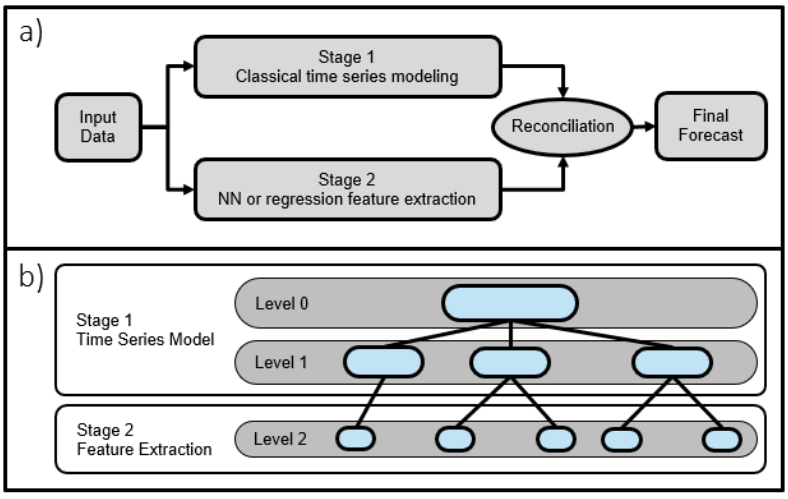

Figure 6. (a) Block Diagram of Multistage Modeling Strategy and (b) Example Division of Hierarchy Levels

WHEN SHOULD YOU USE A NEURAL NETWORK MODEL?

Neural networks are not an all-purpose magic tool to replace classical models (Zhang 2003). To quote George Box, "All models are wrong, but some are useful" (Box and Draper 1987).

For a neural network to be useful, you need more training data than classical time series models require, and you need independent variables that have complex nonlinear relationships to the dependent variable (Box 1976, Yoshio, Hipel and Mcleod 2005, Crone and Häger 2016)

Typical data for which neural networks work well have the following characteristics:

- Large data volume, such as a panel of time series
- Historical data length of at least 300 time ID values
- At least three independent variables, which ideally have nonlinear relationships to the target
- Few (or zero!) missing values

QUICK-START SETTINGS

If you just want to jump in and start training some models, then the tables in this section offer a good starting point for the PSNN and stacked models. The multistage model uses similar settings if neural networks are chosen to model the lower levels.

Feature Extraction	
Dependent lags	Max(p) from AR(p)
Independent lags	Equal to dependent lags
Seasonal dummies	Yes
ESM forecast	Choose one
Linear trend	
Damped trend	

Model Initialization	
Input standardization	Z-score
Number of hidden layers	1
Number of neurons	10
Hidden layer activation function	Rectifier
Direct connections	No
Dependent var transformation	None
Dependent var standardization	Z-score
Error function	Normal
Output layer activation function	Identity
Neuron connection distribution	Xavier

Model Training	
Algorithm	LBFGS
Number of tries	10
Max training iterations	300
Max time	10
L1 regularization	0
L2 regularization	0
Enable early stopping	Yes
Stagnation limit	10
Enable Autotune	No

Table 1. Quick-Start Settings for PSNN and Stacked Models

CASE STUDY

DATA DESCRIPTION

The data set that is used in this example was obtained from Chicago's Array of Things (AoT) project. The data consist of one month of sensor output from modules that are placed around the city to measure air quality, light, and other environmental conditions.

DATA PREPARATION

Cleaning the data is an important first step to ensure that the model can make accurate predictions. This process is outside the scope of this paper, but the code can be found here: https://github.com/sascommunities/sas-global-forum-2020/tree/master/papers/4493-2020-Mills. The AoT data had many missing values and broken or malfunctioning sensors.

Columns for irrelevant variables and sensors that were clearly malfunctioning are removed, and the important independent variables are chosen. You might be wondering why you should choose important variables instead of including everything. It is true that NNs require a large amount of data, but the data quality is important too. Extraneous variables and multicollinearity among variables can reduce forecast accuracy and increase the training

time significantly, so unimportant variables should be excluded (Diaconescu 2008, Christ, Kempa-Liehr and and Feindt 2016). Variable importance can be evaluated by using machine learning methods or an ARIMAX model if required, or you can use domain knowledge about ozone formation, which indicates that UV light, temperature, humidity, and the presence of other pollutants (such as nitrogen oxides from combustion engine exhaust) are important factors. Gas sensors require calibration and exhibit signal drift over time, so it would be best not to use other gas sensors to predict the output of the ozone sensor. In this case study, the chosen predictors are temperature, humidity, and infrared, ultraviolet, and visible light.

After cleaning and organizing the data, 27 series of hourly sensor data remain. The series are delineated by the BY variable, and they are accumulated to an hourly interval for a length of 720 observations each. One final step splits the data into two tables: AoT_train and AoT_test. The resulting data set contains eight columns and 19,440 rows. Table 2 describes the variables in the data.

Variable name	Role	Description
Timestamp	Time ID	Timestamp for observation
Sensor_node_id	BY variable	BY variable from original data
O3_concentration	Dependent variable	Ozone sensor output
Si1145_ir_intensity	Independent variable	Infrared light sensor output
Si1145_uv_intensity	Independent variable	Ultraviolet light sensor output
Si1145_visible_light	Independent variable	Visible light sensor output
At1_temperature	Independent variable	Temperature sensor output
Hih4030_humidity	Independent variable	Humidity sensor output

Table 2. Variables in the Input Data Set

FEATURE EXTRACTION

Now that the data are prepared, you are ready to choose the features to extract. The following sections describe how you can select the appropriate number of lags, the seasonal dummy variables, and the type of trend component.

It is important to understand how feature extraction impacts the effective number of input variables and model parameters. For example, generating 3 lags of 10 variables results in a total of 40 variables and 40 nodes in the input layer. If there are 5 hidden nodes in the first hidden layer, then there are 200 connection weight parameters to solve. A NN model can quickly become very complex which causes the training time to increase dramatically.

Lags

Neural network theory assumes that each observation is independent from any other observation. This assumption makes it difficult to learn autocorrelated features that are common in time series, such as trends and seasonality. Generating lags of the dependent and independent variables helps the model understand the local level and trend in a series similarly to the way an ARIMA model uses an autoregressive (AR) term.

It is important to realize how missing values affect the generation of lagged variables. Consider the example shown in Table 3, which consists of six observations. The independent variable, X, has one missing value, and three lags have been generated for both X and Y.

Date	Y	Lag1(Y)	Lag2(Y)	Lag3(Y)	X	Lag1(X)	Lag2(X)	Lag3(X)
1-Jan	16	.	.	.	4	.	.	.
2-Jan	25	16	.	.	6.25	4	.	.
3-Jan	21	25	16	.	.	6.25	4	.
4-Jan	22	21	25	16	5.5	.	6.25	4
5-Jan	14	22	21	25	3.5	5.5	.	6.25
6-Jan	17	14	22	21	4.25	3.5	5.5	.

Table 3. Example of Missing Values Propagating When Lags Are Generated

Notice that the first three observations contain missing values in the lags of the dependent variable (Y) because the historical data before January 1 is not available. Also notice that the missing value for the independent variable (X) on January 3 propagates downward, resulting in missing values for the last three observations. Every observation in this data set has a missing value, so none of them can be used in the neural network model. If your data have many missing values or you use many lags (or both), then you quickly run out of complete observations for training the NN.

Fitting an AR(*p*) model by using the TSMODEL procedure and including independent variables will indicate an appropriate number of lags and show whether a trend component is detected. The code to fit the AR model over the 27 series and generate a histogram of the AR orders can be found here: https://github.com/sascommunities/sas-global-forum-2020/tree/master/papers/4493-2020-Mills. The histogram is shown in Figure 7 and indicates that three lags should be enough. You might notice the total number of series in the histogram is 20 rather than 27. Some of the series could not be fit to an AR(*p*) model and are excluded. This is an important point: The AR(*p*) model indicates the appropriate number of lags, but a small adjustment to that number at the end of the analysis might result in a better model.

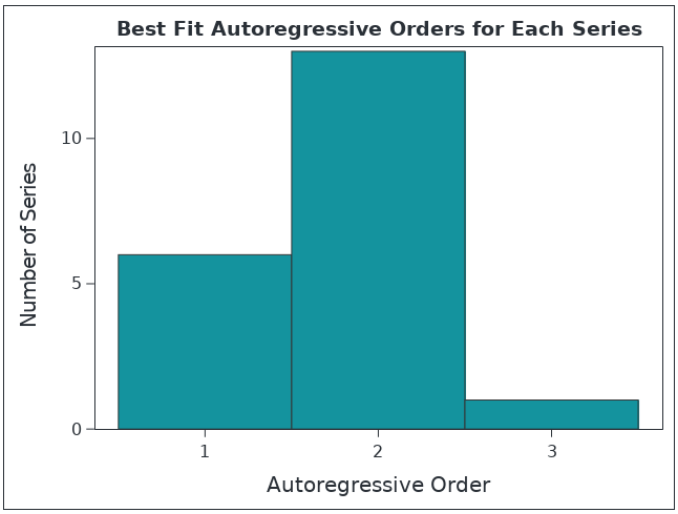

Figure 7. Histogram of Autoregressive Orders Determined by Fitting an AR(*p*) Model with Exogenous Variables

Seasonal Dummy Variables

Generating lags of variables quickly increases the volume of data and the time required to train a neural network. In addition, some data are sacrificed at the beginning of each series as shown previously. Imagine generating a seasonal lag for monthly data, which results in a

whole year of data that can't be used because of missing values. Sacrificing a large portion of the historical data is usually a bad idea.

Seasonal dummy variables can be included to help capture seasonal fluctuations in the data without any data loss. In addition, the seasonality can be specified independently of the time series seasonality. For example, if the time series is accumulated to a weekly interval, then the default behavior would generate 52 seasonal dummy variables to cover the 52 weeks in a year. Specifying "month" or "qtr" as the seasonal dummy interval results in fewer dummy variables and faster convergence when training a neural network.

The AoT data in this example are accumulated to an hourly interval, so 24 seasonal dummy variables are generated, one for each hour of the day. This could be very useful because certain times of the day have increased traffic or sunlight to contribute to ozone generation. Seasonal dummy variables allow capturing information about the hour of the day without creating many lagged variables. Avoiding the use of many lagged variables results in a smaller data table and a faster training process.

Trend Component

A trend component can be extracted from the data to help the model learn. You can choose a linear trend, a damped trend, or an exponential smoothing model (ESM) of the dependent variable to include as an independent variable. The linear and damped trends are special cases of the ESM, so only one of these options should be selected. If an ESM is selected, then the time series forecasting engine chooses the best ESM type on the basis of the training and holdout data partitions. For more control over the extracted model, you can choose between a linear or damped trend on the basis of the length of the holdout and forecast periods. When forecasting further into the future, linear trends often overestimate a series. A damped trend will yield better results.

If you rerun the AR(p) model code on the AoT data without specifying the trend component, then the outModelInfo table shows that just over half of the series have some type of trend. Since you are forecasting out 24 periods, a damped trend is likely the better choice.

This analysis suggests the following features to include in the NN model:

- three lags of salient variables
- hourly seasonal dummy variables
- damped trend component

MODEL INITIALIZATION AND TRAINING

You are now prepared to create a project and generate forecasts using NNs in SAS Visual Forecasting. The following steps walk you through the project creation and configuration:

1. Create a new forecasting project in Visual Forecasting with the AOT_train data set.
2. Go to Project Settings from the gear icon in the upper right corner and change the forecast horizon from 12 to 24.
3. On the Data tab, assign the variable roles as described in Table 2.
4. On the Pipelines tab, delete the Auto-forecasting node if you started with one.
5. Add a Hierarchical Forecasting (Pluggable) node and a PSNN node to the pipeline.
6. Set the Feature Extraction options in the PSNN node. For more information about these features, see the section, "Feature Extraction."
7. Under Model Selection, set Holdout Sample Size to 24 for both modeling nodes.
8. The additional settings for the PSNN nodes are summarized in Table 4.

Feature Extraction	
Dependent lags	3
Independent lags	3
Seasonal dummies	Yes
ESM forecast	No
Linear trend	No
Damped trend	Yes

Model Initialization	
Input standardization	Z-score
Number of hidden layers	1
Number of neurons	10
Hidden layer activation function	Rectifier
Direct connections	No
Dependent variable transformation	None
Dependent variable standardization	Z-score
Error function	Normal
Output layer activation function	Identity
Neuron connection distribution	Xavier

Model Training	
Algorithm	LBFGS
Number of tries	10
Max training iterations	300
Max time	10
L1 regularization	0
L2 regularization	0
Enable early stopping	Yes
Stagnation limit	10
Enable Autotune	No

Table 4. PSNN Settings Used to Fit a Model to the AoT Data

Running the pipeline and viewing the results show that the PSNN performs substantially better than hierarchical forecasting on in-sample data. Keep in mind that the NN training process depends on random initialization and that race conditions in parallel processing threads can also cause variation in output. Therefore, you might see slightly different numbers for the PSNN weighted mean absolute percentage error (WMAPE) measurements.

To view the out-of-sample results, substitute AoT_test for AoT_train on the Data tab, change the Forecasting Task from Diagnose to Forecast for each modeling node, and rerun the pipeline. The WMAPE values are summarized in Table 5, and the last two days of forecasts are plotted in Figure 8 for a representative sensor node. The shaded region in the right half of Figure 8 is the forecast horizon.

Model	In-sample WMAPE	Out-of-sample WMAPE
Hierarchical forecasting	23.0136	10.9508
PSNN	10.9310	9.7429

Table 5. Weighted MAPE Calculated for the PSNN and Hierarchical Models

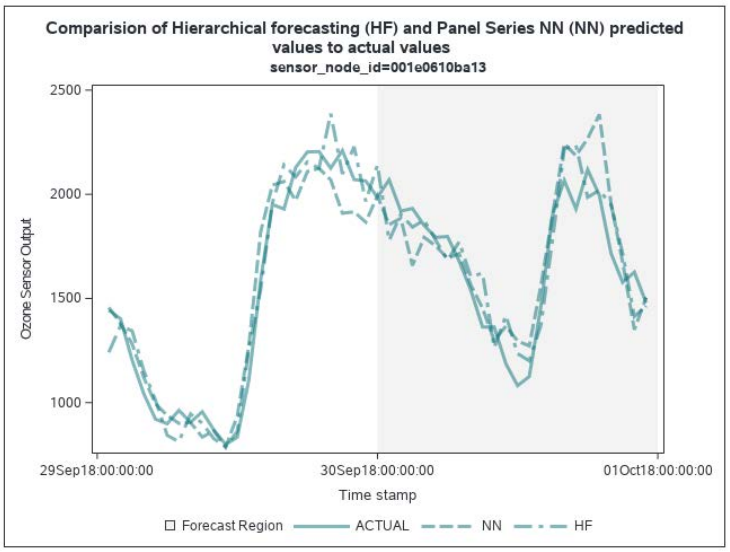

Figure 8. Forecast Comparison between PSNN and Hierarchical Forecasting

To demonstrate the stacked model, create a second pipeline with an Auto-forecasting node and a Stacked Model node. These two nodes do not currently support the Forecast Task or Holdout options, so they are compared against each other based on the in-sample accuracy. Configure the stacked model with the same feature generation and model parameters as were used to configure the PSNN. Then, run the pipeline and view the results. Table 6 shows that the stacked model outperforms auto-forecasting and hierarchical forecasting for this data set.

Model	In-sample WMAPE
Auto-forecasting	22.1265
Hierarchical forecasting	20.8441
Stacked model	10.8316

Table 6. Weighted MAPE Calculated for the Auto-forecasting, Hierarchical, and Stacked Models

SCALING

Training time for neural networks is heavily influenced by the amount of data, the available processing power, and the training specifications. Figure 9 shows the training time dependence for the AoT data as the number of BY groups increase 10×, 50×, and 500×. These additional data are simulated by creating another BY variable and duplicating the original data. The plot displays a nice linear trend as the data size increases.

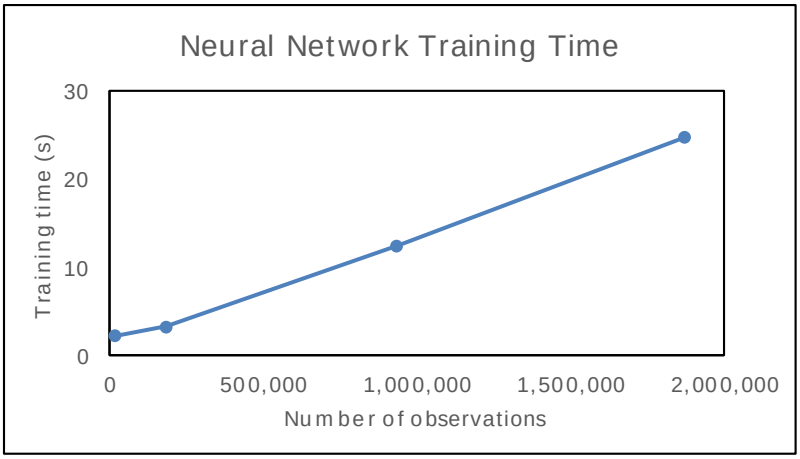

Figure 9. Scaling of Neural Network Training Time with Number of Observations

Scaling the data down provides insight into the minimum amount of data required. The effect of reducing the length of the historical record is shown in Figure 10. Using the full month of data (720 observations in each series) provides the data in Table 5. The WMAPE increases approximately linearly as data are discarded until around 50% (360 observations) remain. Further reduction of the historical record length results in a more drastic increase in WMAPE. Be cautious if you have only a few hundred historical time points to work with.

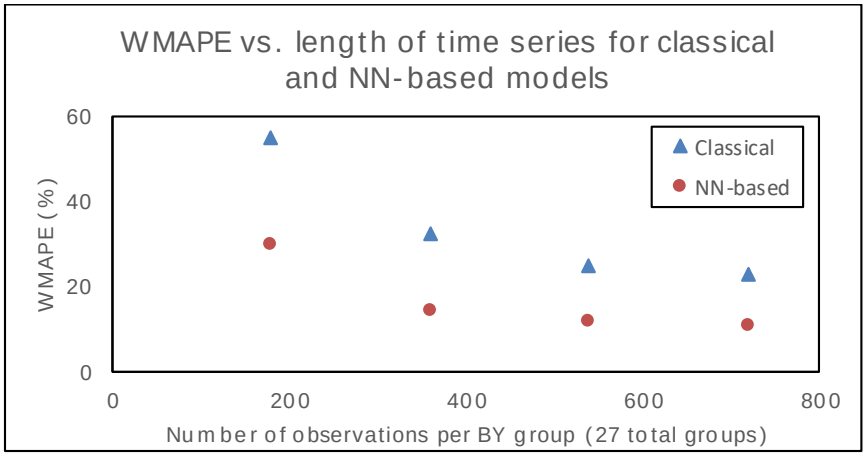

Figure 10. WMAPE as a Function of Historical Record Length for Classical and NN-Based models

Reducing the number of BY groups and holding the series length constant does not necessarily impact the error measurement the same way. Figure 11 shows the lowest error measurement for seven BY groups. Increasing the number of BY groups causes an increase in WMAPE for both classical and NN-based models. Further increasing the number of series shows a slight reduction in WMAPE for NN-based models, whereas the classical models display higher WMAPE with no clear trend. These data indicate that additional BY groups might help reduce forecasting error, but also indicate that a small number of similar series might produce better results. The small number of series has less variation (allowing for a tighter fit), but the model would likely be less accurate if it were used to forecast the other series.

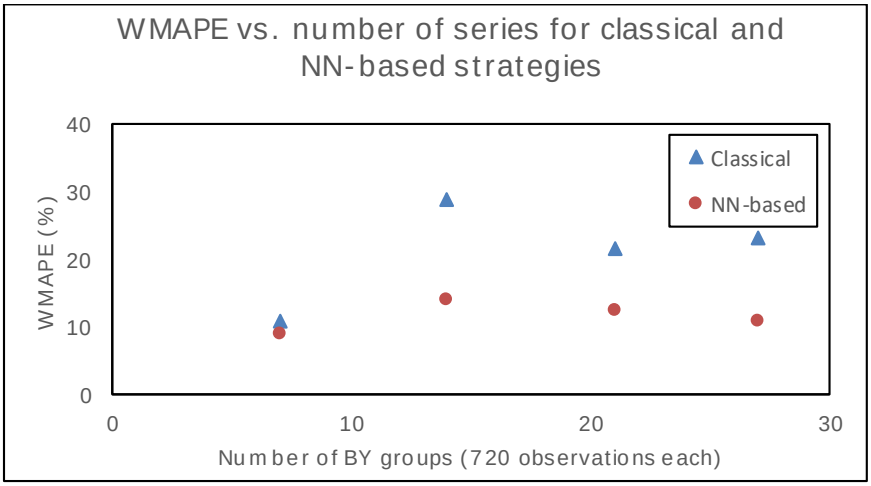

Figure 11. WMAPE as a Function of the Number of BY Groups for Classical and NN-based models

BEST PRACTICES AND OTHER TIPS

Neural network strategies have many parameters to set up and initialize. This section discusses some common pitfalls that are related to the different parameters. Data standardization and activation functions must be chosen along with architectural choices such as the numbers of hidden layers and nodes and the initial neuron connection weights. Model training options such as the optimization algorithm, number of tries, and stopping criteria must also be specified.

OUT-OF-SAMPLE TESTING

Neural networks and other machine learning models are prone to overfitting during the training process. Overfitting causes the models to make very accurate predictions in the training data but generalize poorly to new data. SAS Visual Forecasting includes a robust early-stopping option that does a good job of preventing overfitting, but reserving part of the data for out-of-sample testing is still a good practice. The following code shows how to subset the data easily in SAS Studio and save the data to the "public" caslib before beginning the modeling process:

```
data public.hourly_data_train;
   set public.hourly_data;
   where timestamp lt "30SEP18:00:00"dt;
run;

data public.hourly_data_test;
  set public.hourly_data;
  where timestamp ge "30SEP18:00:00"dt;
run;

proc cas;
    table.save /
        table={name="hourly_data_train", caslib="public"}
        name="hourly_data_train.sas7bdat"
        caslib="public";
    table.save /
        table={name="hourly_data_test", caslib="public"}
        name="hourly_data_test.sas7bdat"
        caslib="public";
run;
quit;
```

NEURAL NETWORK ARCHITECTURE

Choosing the number of hidden layers and neurons is the primary way that you can specify the NN architecture. As in time series models, parsimony is your friend here. Smaller neural networks often perform better than those that have many nodes and layers for two reasons: First, the training time and the number of model parameters increase rapidly with the number of nodes and layers, thus requiring longer times to train. Second, as the network size grows, it is more likely to learn more complex interactions, which eventually leads to overfitting (Diaconescu 2008). It is recommended that you start with a single layer containing 10 nodes and adjust from there if you are not satisfied. Autotuning can also be applied, but it has an even more drastic effect on the training time and is often not necessary. The distribution of connection weights should also be specified. The Xavier and normal distributions are good choices.

ACTIVATION FUNCTIONS AND STANDARDIZATION

Not all activation functions and data standardization methods are compatible with each other. Two options are available for standardization of the input and output data: z-score and mid-range. Z-score standardization scales values to a zero mean and standard deviation equal to 1, whereas mid-range standardization scales values to fall within the range (−1,1). Each activation function also has some range of output values, as shown in Table 7.

The activation function for the output layer must be compatible with the output standardization method. Consider an example that uses the tanh function on the output layer and z-score standardization for the output values. In this case, the output tanh function is trying to map its $(-1,1)$ output to data with a range of $(-\sigma, \sigma)$, which results in loss of information for $|\sigma| > 1$, where σ is the standard deviation.

Activation Function	Input Range	Output Range
sigmoid	[-∞,∞]	(0,1)
logistic	[-∞,∞]	(0,1)
tanh	[-∞,∞]	(-1,1)
rectifier	[-∞,∞]	[0,∞]
identity	[-∞,∞]	[-∞,∞]
sine	[-∞,∞]	(-1,1)

Table 7. Activation Function Ranges

Neural networks for time series generally work well by using a rectifier activation function for hidden layers and an identity activation function on the output layer. This configuration allows any standardization method for the input and output variables. If you try different activation functions, then you must ensure compatibility between the output layer activation function and the output standardization.

ALGORITHM

Two algorithm options are available for training the PSNN and stacked model. Historically, stochastic gradient descent (SGD) has been used often for neural networks, but SGD frequently requires subtle tuning of its parameters such as the learning rate and annealing rate. If your data set is large (over 100,000 rows), then SGD might be a good option. However, for most applications of SAS Visual Forecasting, the limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm converges faster and is easier to use.

OVERFITTING

Regularization is one technique to avoid overfitting because it penalizes large connection weights. L1 and L2 regularization terms can be specified, but they should be used sparingly because incorporating lagged variables into the training data can also have a regularizing effect and lead to an underfitted model. You can start with 0 for both L1 and L2 regularization and try increasing them if your out-of-sample fit is much worse than the in-sample fit.

The early-stopping mechanism is very effective at stopping the training/validation cycle when the validation fit stops improving. However, the error that is measured over the input space is almost always nonconvex, which means that multiple local minima can be found. The stagnation property specifies how many training iterations can complete with no improvement in the validation fit before ending the training/validation cycle. This helps the solver find its way out of local minima so that it can continue searching for better solutions.

CONCLUSION

This paper introduces the neural network-based modeling strategies available in SAS Visual Forecasting and provides guidelines for using them effectively. It explains important considerations for using machine learning strategies for time series forecasting, including structuring of the input data and extraction of features from the input data to aid in

modeling. Guidelines for determining when to use a machine learning model are shown, and an example panel series neural network model was fit to data from Chicago's Array of Things environmental monitoring project. The example was shown to significantly outperform hierarchical forecasting, as evidenced by a WMAPE error measurement about half the size of that for the classical model. Other details and pitfalls not covered in the example are explained, including architecture considerations, regularization, and compatibility of activation functions with data standardization techniques. Finally, the algorithm scaling is evaluated, showing that training time increases linearly with the number of observations.

REFERENCE

Box, George E P. 1976. "Science and Statistics." *Journal of the American Statistical Association* 71 (356): 791-799.

Box, George E P, and Norman R Draper. 1987. *Empirical Model-Building and Response Surfaces.* New York: Wiley.

Christ, Maximilian, Andreas W Kempa-Liehr, and Michael and Feindt. 2016. "Distributed and Parallel Time Series Feature Extraction for Industrial Big Data Applications." *ArXive Pre-Print.* arXiv:1610.07717v3.

Crone, Sven F, and Stephan Häger. 2016. "Feature Selection of Autoregressive Neural Network Inputs for Trend Time Series Forecasting." *IEEE Internations Joint Conference on Neural Networks.*

Diaconescu, Eugen. 2008. "The Use of NARX Neural Networks to Predict Chaotic Time Series." *WSEAS Transactions on Computer Research* 3 (3): 182-191.

Goodfellow, Ian, Bengio Yoshua, and Aaron Courville. 2016. *Back-Propagation and Other Differentiation Algorithms.* Cambridge: The MIT Press.

Taşpınar, F. 2015. "Improving Artificial Neural Network Model Predictions of Daily Average PM10 Concentrations by Applying Principle Component Analysis and Implementing Seasonal Models." *Journal of the Air & Waste Management Association* 65 (7): 800-809.

Yoshio, Kajitani, Kieth W Hipel, and A Ian Mcleod. 2005. "Forecasting Nonlinear Time Series with Feed-Forward Neural Networks: A Case Study of Canadian Lynx Data." *Journal of Forecasting* 24 (2): 105-117.

Zhang, G. 2003. "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model." *Neurocomputing* 50: 159-175.

ACKNOWLEDGMENTS

Special thanks go out to the Array of Things team, the City of Chicago, the National Science Foundation, and any other contributors to the Array of Things project for their hard work and for making the data available to the public.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Steven Mills
Steven.Mills@sas.com