Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Improving forecast stability using deep learning

Jente Van Belle [a,c,*], Ruben Crevits [b], Wouter Verbeke [c]

[a] Data Analytics Laboratory, Solvay Business School, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Belgium
[b] OMP, Koralenhoeve 23, 2160 Wommelgem, Belgium
[c] Faculty of Economics and Business, Katholieke Universiteit Leuven, Naamsestraat 69 box 3555, 3000 Leuven, Belgium

ARTICLE INFO

Keywords:
Forecast accuracy
Forecast instability
Global models
N-BEATS
Regularization

ABSTRACT

In this paper, we define forecast (in)stability in terms of the variability in forecasts for a specific time period caused by updating the forecast for this time period when new observations become available, i.e., as time passes. We propose an extension to the state-of-the-art N-BEATS deep learning architecture for the univariate time series point forecasting problem. The extension allows us to optimize forecasts from both a traditional forecast accuracy perspective as well as a forecast stability perspective. We show that the proposed extension results in forecasts that are more stable without leading to a deterioration in forecast accuracy for the M3 and M4 data sets. Moreover, our experimental study shows that it is possible to improve both forecast accuracy and stability compared to the original N-BEATS architecture, indicating that including a forecast instability component in the loss function can be used as regularization mechanism.

1. Introduction

Traditionally, assessing forecast performance is focused on forecast accuracy. In this paper, we evaluate forecasts from both a forecast accuracy and forecast (in)stability perspective, where the latter concerns variability induced by updating forecasts when new observations become available as time passes. Such forecast updates are considered to result in both benefits and costs. The benefits are caused by an increase in forecast accuracy due to a shorter forecast horizon, while the costs follow from induced forecast instability. In this paper, we aim to improve forecast stability without causing a deterioration in forecast accuracy at the same time. To put it differently, we aim to improve the profitability of forecast updates by decreasing the cost of forecast instability relative to the benefit obtained from an increase in forecast accuracy—i.e., to increase the net benefit or profit from forecast updates by improving forecast stability.

Therefore, we propose an extension to the state-of-the-art N-BEATS deep learning architecture proposed by Oreshkin, Carpov, Chapados, and Bengio (2020) for the univariate time series point forecasting problem, to improve forecast stability by adding a forecast instability component to the loss function. The results of our experimental study show that, for the M3 and M4 competition data sets (Makridakis & Hibon, 2000; Makridakis, Spiliotis, & Assimakopoulos, 2020), our methodology can improve forecast stability without causing a loss in forecast accuracy. Moreover, the results on the validation sets indicate that the proposed methodology can improve both forecast stability and forecast accuracy. Therefore, adding the instability component to the loss function can also be considered an alternative regularization mechanism for global time series models.

We build on the N-BEATS deep learning architecture recently proposed in Oreshkin et al. (2020), which claims to be the first work to empirically demonstrate that a

* Corresponding author at: Faculty of Economics and Business, Katholieke Universiteit Leuven, Naamsestraat 69 box 3555, 3000 Leuven, Belgium.
E-mail addresses: jente.vanbelle@kuleuven.be (J. Van Belle), rcrevits@omp.com (R. Crevits), wouter.verbeke@kuleuven.be (W. Verbeke).

https://doi.org/10.1016/j.ijforecast.2022.06.007
0169-2070/© 2022 International Institute of Forecasters. Published by Elsevier B.V. All rights reserved.

pure deep learning model, using no time-series-specific components, is able to outperform well-established statistical approaches on the M3 and M4 competition data sets. The N-BEATS architecture is used to build a so-called global forecasting model (Januschowski et al., 2020) by training an N-BEATS network with a set of global model parameters across time series. Data-driven global forecasting models tend to be a good choice for operational forecasting problems when they are trained on a large number of time series (Januschowski, Kolassa, et al., 2019). For a detailed discussion of the advantages and disadvantages of global (versus local) forecasting models, see Januschowski et al. (2020).

The remainder of this paper is organized as follows: In Section 2, we review related work on forecast (in)stability and we present an illustrative example to clarify the concept of forecast (in)stability and the important implications that it entails. In Section 3, we first briefly explain the original N-BEATS deep learning architecture before presenting the methodology to improve the forecast stability of N-BEATS forecasts. Section 4 introduces an experimental study and presents the results. Finally, Section 5 concludes the paper and presents directions for future research.

2. Forecast (in)stability

In this section, we first provide a brief overview of the literature on forecast (in)stability. Based on this overview, it is clear that there are several definitions in use for the term "forecast (in)stability". Therefore, we subsequently present an illustrative example to clarify what exactly we mean by forecast (in)stability in this work, and what the implications and their importance are in the context of demand forecasting for supply chain planning, among other applications.

2.1. Related literature

The term "forecast (in)stability" mainly occurs in the literature on model selection uncertainty and model combinations. Model selection uncertainty refers to the uncertainty associated with choosing a single best model among a variety of candidate models. If, for some models, the values of the model selection criterion are similar, a slight change of the data (e.g., by adding random noise) may result in selecting a different model, which potentially yields a very different forecast. In this stream of literature, the term "forecast instability" is thus used to refer to *model selection forecast instability*, i.e., differences in forecasts for the same period due to the selection of different models. Combining models to deal with instability in model selection has been recognized as a possible solution in the literature on statistics and machine learning (see, e.g., Breiman, 1996; Yuan & Yang, 2005) as well as in the forecasting literature (see, e.g., Chatfield, 1996; Kolassa, 2011; Zou & Yang, 2004), where it is generally accepted that combining forecasts into a single ensemble forecast often leads to improvements in forecast accuracy, especially when there is substantial uncertainty in identifying the "best" model. For a discussion on measuring and evaluating model selection uncertainty, see Zou and Yang (2004).

In this paper, we use the term "forecast (in)stability" to refer to a different source of instability, i.e., *rolling origin forecast instability*. Rather than resulting from the selection of different models, this type of instability concerns the variability in forecasts for a specific time period caused by updating the forecast for this time period each time a new observation becomes available, or in other words, from using subsequent forecasting origins (i.e., the time period from which the forecast is generated). This definition of forecast (in)stability is used in, for example, the literature on supply chain planning (see, e.g., Schuster, Ehm, Hottenrott, & Lauer, 2017), in which forecast updating in a rolling fashion, as described above, is common practice. In a supply chain planning context, forecast instability results in so-called system nervousness, which refers to the need to revise supply plans to realign planned supply with forecasted demand and therefore gives rise to supply chain costs (Li & Disney, 2017; Tunc, Kilic, Tarim, & Eksioglu, 2013). The literature on the problem of rolling origin forecast instability in a supply chain planning context is scarce, probably due to the widely accepted fact that it is difficult to directly quantify the costs associated with the system nervousness to which it gives rise (Tunc et al., 2013). However, it is exactly this rolling origin forecast instability that we aim to reduce in this work. A second forecasting application in which this type of forecast instability can be of interest is macroeconomic forecasting, where forecasts are used to decide on new policies and to adjust existing ones. As is the case in a supply chain planning context, in a macroeconomic forecasting context, it is hard to quantify the costs associated with forecast instability. Moreover, for macroeconomic forecasting, in addition to new observations, data revisions have an impact on forecast updating and instability. Dealing with these so-called data vintages further complicates the problem. Combining forecasts has been proposed to deal with forecast instability in this macroeconomic forecasting context as well (see, e.g., Altavilla & Ciccarelli, 2007).

To gain a better understanding of the general problem of rolling origin forecast instability, which may be of interest in all settings in which forecasts are updated and forecast instability comes with certain costs, a detailed illustrative example is provided in the next section.

2.2. Illustrative example of rolling origin forecast instability

Consider the time series depicted by the black line in Fig. 1, which represents 42 observations of the monthly time series N1979 from the M3 competition data set (Makridakis & Hibon, 2000). Assume that we need to produce one- to six-step-ahead forecasts for this time series in every period, and that each forecast horizon serves a different purpose. For instance, in a supply chain planning context, a longer-horizon (e.g., six months ahead) forecast may be needed as input to decide how much of a specific raw material should be ordered (possibly after grouping the forecasted demand of multiple products),

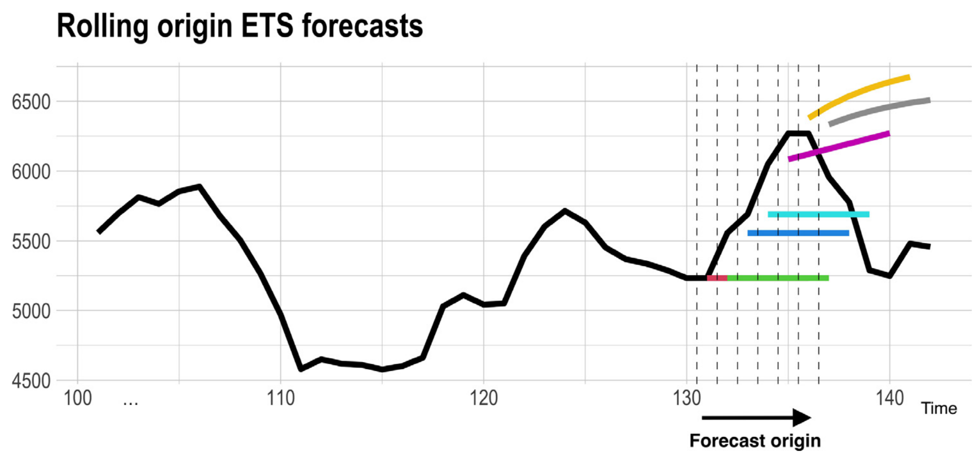


Fig. 1. Rolling origin one- to six-step-ahead ets forecasts for example time series. Different colors are used to visualize forecasts made at different forecasting origins. (For the color version of this figure, the reader is referred to the web version of this article.)

while a shorter-horizon (e.g., three months ahead) forecast may be needed to determine how much to produce of a specific stock keeping unit.

To produce one- to six-step-ahead forecasts in every period, we use the well-known `ets(model = "ZZZ")` function from the `forecast` R package (Hyndman et al., 2020; Hyndman & Khandakar, 2008). With the argument `model = "ZZZ"`, this function relies on state-space formulations of the exponential smoothing methods to automatically select the ETS model type, i.e., the types of error, trend, and seasonal components, using information criteria such as the corrected Akaike information criterion (Hyndman, Koehler, Ord, & Snyder, 2008; Hyndman, Koehler, Snyder, & Grose, 2002). The first letter refers to the error type and can be either `"A"` for additive or `"M"` for multiplicative. The second letter denotes the trend type and can be either `"A"`, `"Ad"` for damped additive, or `"N"` for none. And the third letter denotes the season type and can be either `"A"`, `"M"`, or `"N"`. We use a rolling forecasting origin procedure with an expanding window for the training data (Tashman, 2000). In Fig. 1, the `ets` forecasts resulting from a different forecasting origin are represented by different colors.

By inspecting the different forecasts in Fig. 1, it seems that the forecasts for a specific time period become more accurate when approaching this period. This observation is confirmed by evaluating the accuracy of these forecasts with the symmetric mean absolute percentage error (sMAPE), a scale-independent error metric used in the M3 and M4 competitions (Makridakis & Hibon, 2000; Makridakis et al., 2020). The sMAPE across one- to h-step-ahead forecasts resulting from a specific forecasting origin t is obtained as follows:

$$\text{sMAPE} = \frac{200}{h}\sum_{i=1}^{h}\frac{|y_{t+i} - \hat{y}_{t+i|t}|}{|y_{t+i}| + |\hat{y}_{t+i|t}|} \quad (1)$$

where y_{t+i} is the observed value of the examined time series at time $t + i$, $\hat{y}_{t+i|t}$ is the forecast for period $t + i$ made at time t, i.e., the forecasting origin, and h is the maximum forecast horizon. To obtain the performance across different forecasting origins, we simply take the average of the corresponding sMAPE values. The top row in Table 1 summarizes forecast accuracy per forecast horizon for this forecast updating scenario and shows the stylized relationship that exists between forecast accuracy and forecast horizon: the further that we have to predict into the future, the larger the forecast error is on average.

However, we can also look at the forecasts in Fig. 1 from a different perspective, namely, a forecast stability perspective. While a forecast for a specific time period typically becomes more accurate when approaching this period, from Fig. 1 it is also obvious that updating the forecast for a specific period when a new data point becomes available potentially results in large changes to the previously made forecasts for that period. Given that we are dealing with `ets` forecasts here, this forecast instability is due to the effect of the newly available observations on either parameter estimation or a combination of model selection and parameter estimation. For instance, the differences in the forecasts originating from the last two forecasting origins, visualized in Fig. 1 by the yellow and grey lines, are due to parameter estimation only, since the forecasts all result from the same ETS model type, i.e., `"MAdN"`. The differences in the forecasts visualized by the cyan and purple lines for the two forecasting origins coming just before, on the other hand, are due to a combination of parameter estimation and ETS model type selection, as they originate from an `"MNN"` model with no trend and an `"MAN"` model with an additive trend, respectively.

Rolling origin forecast instability (which possibly includes model selection forecast instability, as illustrated above) can incur certain costs. For instance, in the context of supply chain planning, this type of forecast instability leads to costs as initial supply plans need to be revised, resulting in building up excessive inventory or expediting the production and/or delivery process at a higher cost (Li & Disney, 2017; Tunc et al., 2013). To measure rolling origin forecast instability, we propose a metric similar to the sMAPE, called the symmetric mean absolute percentage change (sMAPC). For forecasting origins $t-1$ and t, the

Table 1
sMAPE and sMAPC for one- to six-step-ahead `ets` forecasts for example time series across the different forecasting origins used to generate Fig. 1 (i.e., the updating scenario) and the scenario in which forecasts are not updated and only the forecasts made at the first and last forecasting origins are considered.

	Horizon h	1	2	3	4	5	6	Overall
Updating	sMAPE	3.63	7.60	11.39	14.12	14.45	13.89	10.85
	sMAPC	3.55	3.82	4.03	4.20	4.34	–	3.99
No updating	sMAPE	3.08	8.02	13.92	17.61	17.45	17.82	12.98
	sMAPC	0.00	0.00	0.00	0.00	0.00	–	0.00

sMAPC across one- to h-step-ahead forecasts is obtained as:

$$\mathrm{sMAPC} = \frac{200}{(h-1)} \sum_{i=1}^{h-1} \frac{|\hat{y}_{t+i|t} - \hat{y}_{t+i|t-1}|}{|\hat{y}_{t+i|t}| + |\hat{y}_{t+i|t-1}|}. \qquad (2)$$

The difference between the sMAPC and the sMAPE is that the former compares forecasts to forecasts whereas the latter compares forecasts to observed values. While the sMAPC is defined here in terms of two adjacent forecasting origins, $t-1$ and t, it can also be calculated for pairs of nonadjacent forecasting origins. In this work, however, we only consider pairs of adjacent forecasting origins to quantify instability. The instability across different pairs of forecasting origins can be obtained by averaging their corresponding sMAPC values. From the forecasts shown in Fig. 1, an sMAPC of 3.99 is obtained. The second row in Table 1 shows that forecast instability also increases with increasing forecast horizon. In other words, the size by which a longer-horizon forecast changes due to forecast updating is larger on average compared to changes in shorter-horizon forecasts.

In the above example, updating forecasts thus results in both benefits and costs caused by more accurate forecasts and induced forecast instability, respectively, with an sMAPE of 10.85 and an sMAPC of 3.99 across forecast horizons. Comparing these values with a scenario in which we do not update the forecasts, i.e., in which we rely on the forecasts produced at time $t = 130$ and 136, irrespective of the period we are in, illustrates the possible tradeoff between forecast accuracy and forecast stability: as shown in the bottom two rows in Table 1, for these two forecasting origins, the sMAPE and sMAPC are equal to 12.98 and 0, respectively. If forecasts are not updated, we do not incur any additional costs due to induced forecast instability; however, we also cannot benefit from potential improvements in forecast accuracy. Finding a good balance between forecast accuracy and forecast stability by approaching this problem as a tradeoff requires full quantification of their associated costs and benefits, which is often difficult in practice (as is the case for the provided examples of macroeconomic forecasting and demand forecasting for supply chain planning). However, the application of forecast updating in practice implicitly assumes that the induced costs are outweighed by the benefits for the forecasting problem at hand. Therefore, in this paper, we do not approach the relation between forecast accuracy and forecast stability from a tradeoff perspective, but instead focus on assessing whether we can improve forecast stability while maintaining forecast accuracy.

3. A methodology to improve forecast stability using deep learning

In this section, we propose a methodology to improve forecast stability using deep learning. To this end, we build on N-BEATS, a deep learning architecture proposed by Oreshkin et al. (2020) for the univariate time series point forecasting problem. Therefore, we first briefly explain the original N-BEATS architecture.

3.1. Original N-BEATS architecture

Oreshkin et al. (2020) demonstrate the state-of-the-art performance of N-BEATS for all the data sets considered in their study, including the M3 and M4 data sets, and for both configurations of the model presented: a generic and an interpretable configuration. While the interpretable configuration relies on time-series-specific components (trend and seasonality) to achieve interpretability without considerable loss in accuracy, the generic configuration does not and can therefore be considered a *pure* deep learning model. The state-of-the-art performance of the generic model suggests that pure deep learning models can compete with and even outperform hybrid methods such as the best-performing method in M4 (Smyl, 2020) that combines recurrent neural networks with exponential smoothing. In this paper, we therefore focus on the generic configuration. Below, we briefly discuss this generic N-BEATS architecture.

N-BEATS takes as input a vector of T historical observations, also referred to as the lookback window, $\mathbf{x}_{T|t} \in \mathbb{R}^T = [y_{t-T+1}, \ldots, y_t]$ with length $T \leq t$ and forecasting origin t and outputs a prediction $\hat{\mathbf{y}}_{h|t} \in \mathbb{R}^h = [\hat{y}_{t+1|t}, \ldots, \hat{y}_{t+h|t}]$ for the vector $\mathbf{y}_{h|t} \in \mathbb{R}^h = [y_{t+1}, \ldots, y_{t+h}]$ which comprises the next h observations. The length T of the lookback window $\mathbf{x}_{T|t}$ is set to a multiple of the forecast horizon h, with lengths ranging from $2h$ to $7h$ in Oreshkin et al. (2020). The generic N-BEATS architecture, visualized in Fig. 2, is composed of multiple sequentially connected processing blocks $k = 1, \ldots, K$, with the number of blocks K being a hyperparameter. Each block k takes $\mathbf{x}_{T|t}^{[k]}$ as input to produce both a partial forecast $\hat{\mathbf{y}}_{h|t}^{[k]}$ and a backcast $\hat{\mathbf{x}}_{T|t}^{[k]}$ by going through a stack of four fully connected layers with ReLu nonlinearity and a single task-specific layer for each block output, i.e., the backcast and (partial) forecast. The difference between the block input $\mathbf{x}_{T|t}^{[k]}$ and the backcast $\hat{\mathbf{x}}_{T|t}^{[k]}$ forms the input of the next block in the network (with $\mathbf{x}_{T|t}^{[1]} = \mathbf{x}_{T|t}$). The final forecast $\hat{\mathbf{y}}_{h|t}$ is the sum of the (partial) block forecasts. The authors refer to this topology as *doubly residual stacking*:

$$\mathbf{x}_{T|t}^{[k]} = \mathbf{x}_{T|t}^{[k-1]} - \hat{\mathbf{x}}_{T|t}^{[k-1]}, \qquad (3)$$

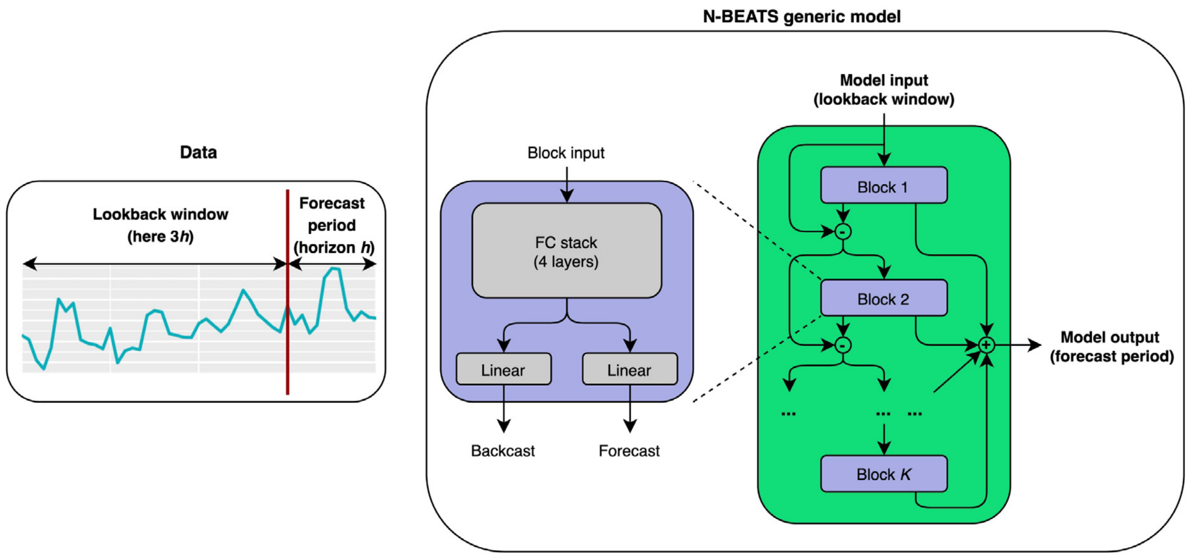


Fig. 2. N-BEATS architecture: generic configuration (Oreshkin et al., 2020).

$$\hat{\mathbf{y}}_{h|t} = \sum_{k=1}^{K} \hat{\mathbf{y}}_{h|t}^{[k]}. \tag{4}$$

The (residual) backcast in each block serves to gradually help the learning in the next blocks by removing the part of the input that is not helpful (anymore) for forecasting. In this way, the raw input signal $\mathbf{x}_{T|t}$ is sequentially analyzed. The authors demonstrate the effectiveness of the proposed doubly residual topology through an extensive ablation study, suggesting that the proposed architecture provides gain that cannot be achieved by simply increasing the number of parameters, and they show that the network architecture is suitable to make forecasts for different frequencies without modification. In what follows, block-level indexing is omitted for the sake of clarity.

To optimize the parameters $\mathbf{W}$ of an N-BEATS network $f(\mathbf{x}_{T|t}; \mathbf{W})$, one relies on a training set $D = \{\mathbf{x}^j_{T|t}, \mathbf{y}^j_{h|t}\}$, obtained by sampling input–output windows across time series and forecasting origins, and a certain loss function L that quantifies forecast errors:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_j L(\mathbf{y}^j_{h|t}, \hat{\mathbf{y}}^j_{h|t}); \qquad \hat{\mathbf{y}}^j_{h|t} = f(\mathbf{x}^j_{T|t}; \mathbf{W}). \tag{5}$$

The expression above is numerically optimized via stochastic gradient descent (this is discussed in more detail in Section 4.1 below). Note that the training sample index j captures the sampling across time series as well as across the available time periods of length $T + h$ for a selected time series. The index t is thus relative and is used to refer to the forecasting origin, irrespective of the training sample's absolute time period (and thus absolute forecasting origin). From the above expression, it is clear that N-BEATS is a global model (Januschowski et al., 2020), as the network parameters are optimized across many time series by minimizing a selected loss function. Therefore, in the absence of data preprocessing, it is necessary to use scale-independent loss functions such as the sMAPE (see Eq. (1)) or the root mean squared scaled error (RMSSE), which is a variant of the well-known mean absolute scaled error (MASE) proposed by Hyndman and Koehler (2006). The RMSSE for a specific input–output sample j with forecasting origin t can be calculated as follows:

$$\text{RMSSE} = \sqrt{\frac{\frac{1}{h}\sum_{i=1}^{h}(y_{t+i} - \hat{y}_{t+i|t})^2}{\frac{1}{(T-1)}\sum_{i=1}^{T-1}(y_{t-i+1} - y_{t-i})^2}}. \tag{6}$$

Finally, to report results on the test sets, Oreshkin et al. (2020) rely on ensembling as a regularization mechanism. They build an ensemble of 180 N-BEATS networks and use the median as the ensemble aggregation function. To obtain 180 different networks, they rely on several sources of diversity: three different loss functions are used for model fitting (MAPE, sMAPE, and MASE), six different lookback window lengths are used ($T = 2h, \ldots, 7h$), and ten different random initializations are used for each of the above combinations, affecting both network parameter initialization and the generation of training batch sequences.

3.2. Extending N-BEATS to improve forecast stability

To improve the forecast stability of N-BEATS forecasts, we propose an extension to the loss function that is used. Note that the proposed extension can also be implemented within other deep learning methods for univariate time series point forecasting.

By considering the overlapping time periods in $\hat{\mathbf{y}}^j_{h|t}$ and $\hat{\mathbf{y}}^j_{h|t-1}$, which contain the forecasts produced by the network for input–output sample j for forecasting origins t and $t-1$, respectively, we can add a forecast instability component to the loss function from Eq. (5) by minimizing the differences between the model forecasts made at time t and $t-1$ for the same periods. That is, we only consider the overlapping time periods $t+1$ to $t+h-1$. Using a (second) lagged input–output pair for the input–output sample j, with the forecasting

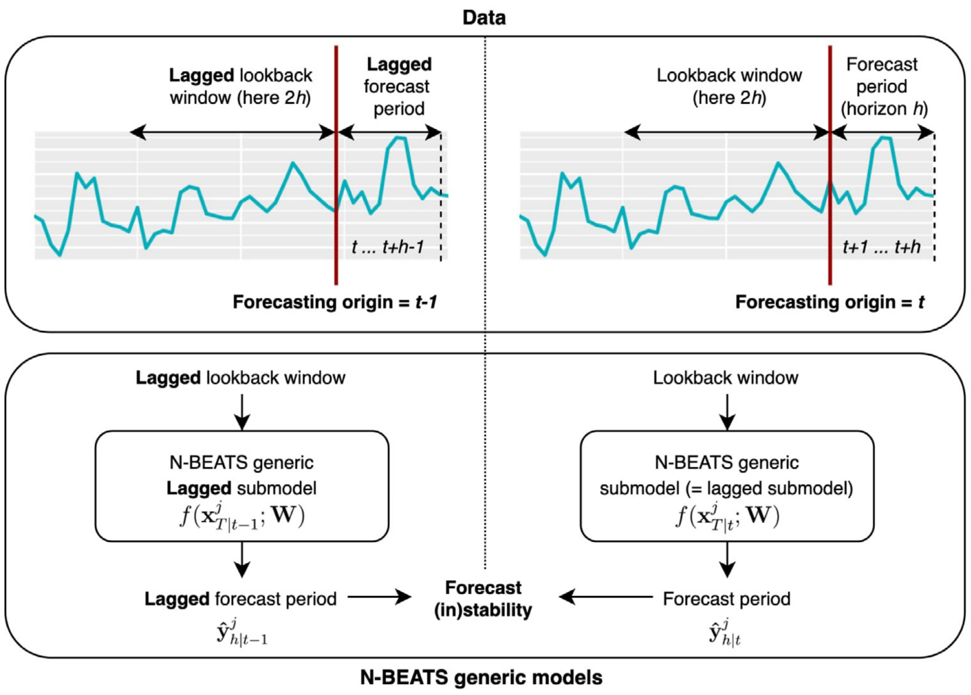

Data
Lagged lookback window (here 2h)
Lagged forecast period
t ... t+h-1
Forecasting origin = t-1
Lookback window (here 2h)
Forecast period (horizon h)
t+1 ... t+h
Forecasting origin = t
Lagged lookback window
N-BEATS generic Lagged submodel $f(\mathbf{x}^j_{T|t-1};\mathbf{W})$
Lagged forecast period $\hat{\mathbf{y}}^j_{h|t-1}$
Forecast (in)stability
Lookback window
N-BEATS generic submodel (= lagged submodel) $f(\mathbf{x}^j_{T|t};\mathbf{W})$
Forecast period $\hat{\mathbf{y}}^j_{h|t}$
N-BEATS generic models

Fig. 3. N-BEATS extension to improve forecast stability.

origin moved one period backward in time, thus allows us to augment the optimization problem with a forecast instability component $\Omega(\hat{\mathbf{y}}^j_{h|t-1}, \hat{\mathbf{y}}^j_{h|t})$:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_j \left(\frac{1}{2}\left(L(\mathbf{y}^j_{h|t}, \hat{\mathbf{y}}^j_{h|t}) + L(\mathbf{y}^j_{h|t-1}, \hat{\mathbf{y}}^j_{h|t-1})\right) + \lambda\Omega(\hat{\mathbf{y}}^j_{h|t-1}, \hat{\mathbf{y}}^j_{h|t})\right), \tag{7}$$

$$\hat{\mathbf{y}}^j_{h|t} = f(\mathbf{x}^j_{T|t}; \mathbf{W}), \tag{8}$$

$$\hat{\mathbf{y}}^j_{h|t-1} = f(\mathbf{x}^j_{T|t-1}; \mathbf{W}), \tag{9}$$

where $\lambda \geq 0$ controls the extent to which the forecasts produced by the network are influenced by the forecast instability component, and Ω is a scale-independent function that quantifies forecast instability such as the sMAPC (see Eq. (2)) or the root mean squared scaled change (RMSSC). Along the lines of the RMSSE, we define the RMSSC for a specific input–output sample j and forecasting origins $t-1$ and t as follows:

$$\text{RMSSC} = \sqrt{\frac{\frac{1}{(h-1)}\sum_{i=1}^{h-1}(\hat{y}_{t+i|t} - \hat{y}_{t+i|t-1})^2}{\frac{1}{(T-1)}\sum_{i=1}^{T-1}(y_{t-i+1} - y_{t-i})^2}}. \tag{10}$$

A visualization of the extended N-BEATS architecture is provided in Fig. 3.

Whereas the sole objective in N-BEATS (see Eq. (5)) is to minimize some function of forecast error, here the objective is to minimize forecast error as well as forecast instability. Recall from Section 2.2 that if forecasts are not updated when new observations become available, we do not incur additional costs due to induced forecast instability; however, we also cannot benefit from potential improvements in forecast accuracy. As it is generally accepted that the benefits of forecast updating in terms of forecast accuracy improvements exceed the extra costs induced by forecast instability, generation and evaluation of point forecasts traditionally focus on forecast accuracy. However, we hypothesize that the focus on forecast accuracy does not necessarily imply ignoring forecast instability. Specifically, in the experimental study below, we assess whether there exists a certain range of λ-values that lead to improvements in forecast stability without causing a deterioration in forecast accuracy at the same time.

4. Experimental study

In this section, we present and report the results of experiments carried out to assess the performance of the proposed extension to the N-BEATS architecture in terms of its ability to improve the forecast stability of N-BEATS forecasts while maintaining forecast accuracy. We first provide a detailed description of the experimental design, including the data sets that are used and the evaluation schemes that are adopted, and we give an overview of the forecasting methods and training methodology that are used. Subsequently, experimental results are reported and discussed.

4.1. Experimental design

Data sets. We use the M3 (Makridakis & Hibon, 2000) and M4 (Makridakis et al., 2020) data sets in our experiments. Table 2 provides summary statistics of the data sets included in the experimental study. All series have positive observed values at all time steps.

Table 2
Summary statistics of the data sets used. The third line of the header row contains values for the forecast horizon/the number of forecasting origins used, and the required forecast horizon in the original competition(s) is shown between brackets.

	M3				M4					
	Yearly	Quarterly	Monthly	Other	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
	2/5 (6)	4/5 (8)	6/13 (18)	2/7 (8)	2/5 (6)	4/5 (8)	6/13 (18)	4/10 (13)	7/8 (14)	24/25 (48)
No. of series	645	756	1,428	174	23,000	24,000	48,000	359	4,227	414
Min. length	20	24	66	71	19	24	60	93	107	748
Max. length	47	72	144	104	841	874	2812	2610	9933	1008
Mean length	28.4	48.9	117.3	76.6	37.3	100.2	234.3	1035.0	2371.4	901.9
Std. dev. length	9.9	10.6	28.5	10.9	24.5	51.1	137.4	707.1	1756.6	127.9

Evaluation schemes. For all data sets, standard evaluation schemes were previously defined for use in the original competitions. For instance, for the monthly time series, both competitions required one- to 18-step-ahead forecasts from a single forecasting origin for out-of-sample evaluation; i.e., the last 18 data points of each time series are withheld as a test set. Also, for the other data subsets with different frequencies, only a single forecasting origin is used. However, since we also want to evaluate forecast (in)stability next to forecast accuracy, we cannot readily adopt the standard evaluation schemes, because forecast instability calculations require forecasts from at least two different forecasting origins. Therefore, we adopt different evaluation schemes by using rolling forecasting origin evaluations (Tashman, 2000). The reported results are averages across forecasting origins. For the monthly time series, for instance, we evaluate one- to six-month-ahead forecasts for the test set as in the standard setting. That is, we consider 13 series of one- to six-month-ahead forecasts stemming from 13 different forecasting origins. For the other data subsets with different frequencies, values for the forecast horizon and the number of forecasting origins used, and the forecast horizon required in the original competition(s), are provided in Table 2. Note that the use of these different evaluation schemes implies that we cannot directly compare our results to the results for the M3 and M4 data sets reported in the literature.

Forecasting methods. In addition to reporting the results of standard N-BEATS (see Section 3.1) and the N-BEATS extension to improve forecast stability (see Section 3.2), hereafter referred to as N-BEATS-S, we report the results of the following forecasting methods:

- N-BEATS weight decay: the squared $L2$ norm of the network parameters $\|\mathbf{W}\|_2^2$ is added to the loss function of the standard N-BEATS network. That is, parameter shrinkage (towards zero) is induced to regularize the network.
- N-BEATS dropout: a standard N-BEATS network trained with dropout implemented on all hidden layers to regularize the network (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).
- N-BEATS T ensemble: the median of the forecasts obtained by standard N-BEATS networks with lookback window lengths T equal to (1) a tuned value based on the validation set performance (see Table 3) and (2) $T = 8h$.
- N-BEATS origin ensemble: the mean of all available standard N-BEATS forecasts for a specific period, i.e., reusing forecasts for the time period considered made at previous forecasting origins with longer forecast horizons. For example, for the monthly time series, to obtain a forecast at time t for $t+3$, we combine the produced three-step-ahead forecast with the four- to six-step-ahead forecasts for the same period already produced previously.
- N-BEATS-S weight decay: on top of the forecast instability loss component, the squared $L2$ norm of the network parameters is added to the loss function.
- N-BEATS-S T ensemble: same as above for N-BEATS T ensemble but with N-BEATS-S networks as base models. For each lookback window length T, a suitable λ-value is determined based on the validation set performance.
- N-BEATS-S origin ensemble: same as above for N-BEATS origin ensemble but with N-BEATS-S forecasts as base forecasts.
- ETS: forecasts obtained by using the `ets` function from the `forecast` R package. This exponential smoothing state-space model (with an automated model selection procedure) performed very well in the M3 competition (Hyndman et al., 2008, 2002).
- ARIMA: forecasts obtained by using the `auto.arima` function from the `forecast` R package. This function automatically selects the best ARIMA model based on the algorithm described in Hyndman and Khandakar (2008).
- THETA: forecasts obtained by using the `thetaf` function from the `forecast` R package. The theta method, proposed by Assimakopoulos and Nikolopoulos (2000), was the winning method of the M3 competition (Makridakis & Hibon, 2000).

We run every N-BEATS variant or extension with a specific set of hyperparameters five times, with random initializations. All reported results are obtained by combining forecasts across these runs with the median as the ensemble aggregation function. The N-BEATS(-S), N-BEATS(-S) weight decay, and N-BEATS dropout results are thus all based on five different networks, and the T ensembles on ten. For the origin ensembles, the N-BEATS(-S) forecasts are further combined as described above. Finally, note that for constructing the T ensembles (and to aggregate over different random initializations), we use the median as the ensemble aggregation function, whereas for constructing the origin ensembles we use the mean. The reason for this difference lies in the different motivations to include these methods in this experimental study: the motivation for constructing T ensembles is to obtain more robust and accurate forecasts, and as such, the median is

Table 3
Hyperparameter settings.

		M3				M4					
		Yearly	Quarterly	Monthly	Other	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
N-BEATS	No. of blocks K	20				20					
	Hidden layer width	256				256					
	Batch size	512	512	512	128	512	512	512	256	512	256
	Lookback window length T	$6h$	$6h$	$6h$	$6h$	$4h$	$4h$	$4h$	$4h$	$4h$	$7h$
	Forecasting origin range	$20h$	$20h$	$20h$	$20h$	$20h$	$10h$	$10h$	$20h$	$20h$	$20h$
	Iterations	1000	2000	8000	1500	6600	7130	14,415	2500	12,000	5000
	Learning rate	1e−5	1e−5	1e−5	1e−5	1e−5	1e−3	1e−3	1e−5	1e−5	1e−3
	Weight decay	1e−5	1e−4	2.5e−5	0	0	1e−4	3.16e−6	0	0	1e−5
	Dropout probability	0	0	0	0	0	0.05	0.2	0	0	0
N-BEATS-S	λ	0.053	0.111	0.176	0.081	0.039	0.111	0.176	0.176	0.067	0.333
	Weight decay	1e−4	0	1e−5	0	0	0	1e−5	1e−5	0	0
	λ for $T = 8h$	0.039	0.212	0.481	0.096	0.039	0.176	0.290	0.212	0.067	–

better suited, as more extreme forecasts do not directly affect the ensemble forecast values; the motivation for constructing the origin ensembles, on the other hand, is to obtain more stable forecasts. Although the median could also be used as the ensemble operator, we consider the mean to be a more natural choice for the purpose of smoothing out the effect on forecast instability of forecast updating, since it results in all previous (longer-horizon) forecasts affecting the ensemble forecast values.

Training methodology for N-BEATS networks. Each data set is split into training, validation, and test sets, the latter containing the last n observations of each time series, with n equal to the required forecast horizon in the original competition. We use an equally sized validation set which comprises, for each time series, the n data points that were observed just before the first data point in the test set. For performance evaluation on the validation set, we use the same rolling origin evaluation procedure used to report the test set results. The validation results are used to tune hyperparameters. However, once the hyperparameters are determined, we train the models on the full training set, comprising the training and validation sets, and we report the results on the test set.

An overview of the hyperparameters used for the different subsets of the M3 and M4 data sets is provided in Table 3. Two hyperparameters specific to N-BEATS—namely, the number of blocks K and the width of the hidden layers—and the batch size used to train the N-BEATS networks are determined based on the minimum validation sMAPE resulting from an exploratory study in which the monthly M3 and M4 validation sets are combined. The batch size is adjusted downwards for the subsets containing fewer than 512 series. The other hyperparameters for training standard N-BEATS networks are the lookback window length T, the forecasting origin range, the number of learning iterations, and the learning rate. These hyperparameters are tuned on the data subset level (minimum validation sMAPE and convergence of validation loss for a single run). Note that for the M4-hourly subset, given that $h = 24$ and that there is thus a logical choice for T available in this setting, i.e., $T = 7h$, we only use $T = 7h$ and do not build an N-BEATS-S T ensemble.

For all N-BEATS variants and extensions included in this experimental study, only the additional hyperparameters are tuned, while the common hyperparameters are kept fixed at the (tuned) values for the standard N-BEATS network. For N-BEATS-S, a crucial hyperparameter is λ, as it controls the extent to which the forecast instability component affects the optimization problem and therefore the model outputs. λ is determined on the data subset level (and per lookback window length T) by minimizing the average validation sMAPE obtained across three different runs with random initializations. Validation results for varying λ-values are discussed in more detail for the monthly subsets in Section 4.3 below.

To train the N-BEATS networks, we create training batches with a fixed size which depends on the data subset considered (see batch size in Table 3). As discussed above in Section 3.1, training batches are randomized in two ways: first, time series are sampled uniformly at random with replacement; and second, for each selected time series, a time step is sampled uniformly at random from the forecasting origin range to create input–output windows that can be used for training. The forecasting origin range used to create input–output windows in all cases comprises the most recent range available from which samples can be created without giving rise to missing values. A larger forecasting origin range serves as a counterweight to a smaller number of time series. Finally, note that, to ensure a fair comparison between the N-BEATS and N-BEATS-S networks, we also feed the lagged input–output windows to the standard N-BEATS networks; however, the forecast instability component is ignored by setting $\lambda = 0$.

All networks are implemented and trained in PyTorch (Paszke et al., 2019). For the implementation of a standard N-BEATS network, we followed the description in Oreshkin et al. (2020) as closely as possible and used an available PyTorch implementation[1] as a starting point. Our N-BEATS-S code is available online at https://github.com/KU-Leuven-LIRIS/n-beats-s. We use the Adam optimizer with default settings (Kingma & Ba, 2014) and initial learning rates as specified in Table 3. While forecast accuracy and stability results are reported in terms of sMAPE and sMAPC in the next section (because they are convenient to display and interpret), for training, we use RMSSE and RMSSC, since the use of sMAPE makes

[1] https://github.com/philipperemy/n-beats

Table 4
Forecast accuracy and stability performance on the M3 test sets. Lower values are better. The minimum value per column is highlighted in bold.

	Yearly		Quarterly		Monthly		Other	
	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC
N-BEATS	10.81	8.42	7.34	4.01	11.45	3.67	2.73	2.51
N-BEATS weight decay	10.93	8.22	7.34	3.82	11.47	3.59	–	–
N-BEATS dropout	–	–	–	–	–	–	–	–
N-BEATS T ensemble	**10.78**	8.25	7.25	3.81	11.39	3.72	2.68	2.39
N-BEATS origin ensemble	11.36	4.18	7.74	1.79	11.59	1.30	2.87	1.33
N-BEATS-S	10.85	7.93	7.37	3.61	11.38	2.62	2.73	2.26
N-BEATS-S weight decay	11.19	7.25	–	–	11.41	2.57	–	–
N-BEATS-S T ensemble	10.79	7.78	7.33	3.32	11.36	2.22	2.69	2.13
N-BEATS-S origin ensemble	11.42	**3.94**	7.78	**1.63**	11.59	**1.01**	2.88	**1.20**
ETS	11.30	8.09	7.04	4.07	11.34	3.21	**2.59**	1.88
ARIMA	11.29	9.11	7.29	4.31	11.70	3.16	**2.59**	2.11
THETA	11.25	7.35	**7.00**	3.87	**11.28**	2.96	**2.59**	1.84

Table 5
Forecast accuracy and stability performance on the M4 test sets. Lower values are better. The minimum value per column is highlighted in bold.

	Yearly		Quarterly		Monthly		Weekly		Daily		Hourly	
	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC	sMAPE	sMAPC
N-BEATS	8.14	7.99	7.81	4.31	9.11	3.83	6.63	4.28	**2.07**	0.87	9.26	3.76
N-BEATS weight decay	–	–	7.80	4.27	9.12	3.80	–	–	–	–	9.05	3.53
N-BEATS dropout	–	–	7.80	4.32	9.19	3.89	–	–	–	–	–	–
N-BEATS T ensemble	8.06	7.86	**7.74**	4.18	**8.95**	3.51	6.45	4.04	2.08	0.83	–	–
N-BEATS origin ensemble	8.53	4.04	8.18	1.80	9.63	1.32	6.81	1.65	2.31	0.34	9.02	0.67
N-BEATS-S	8.13	7.54	7.83	3.61	9.11	2.83	6.56	3.17	2.10	0.81	8.93	2.94
N-BEATS-S weight decay	–	–	–	–	9.12	2.78	6.60	3.13	–	–	–	–
N-BEATS-S T ensemble	**8.04**	7.42	7.75	3.42	9.01	2.48	6.42	2.93	2.10	0.78	–	–
N-BEATS-S origin ensemble	8.55	**3.81**	8.25	**1.56**	9.70	**1.06**	6.89	**1.30**	2.34	**0.32**	**8.86**	**0.55**
ETS	8.70	7.07	8.07	4.37	9.98	4.37	6.51	3.26	2.15	1.12	14.96	6.53
ARIMA	8.77	7.71	8.22	4.49	9.78	4.15	**6.37**	3.06	2.21	1.18	26.89	6.63
THETA	8.72	6.38	8.13	4.12	10.07	3.80	6.79	3.29	2.09	0.94	14.71	5.63

training numerically unstable. Moreover, from a forecast stability perspective, the use of a quadratic loss function to quantify forecast instability is intuitive, since high levels of forecast instability are particularly undesirable and should therefore be assigned a relatively high cost. Finally, although no data preprocessing is required as we rely on these scale-independent loss functions, for the M3-other, M4-yearly, and M4-weekly subsets, networks are built by relying on standardized time series to avoid numerical instabilities.

4.2. Results

Tables 4 and 5 summarize the test set results for the M3 and M4 data sets, respectively. Performance measures are first computed for each time series separately by averaging their values across forecasting origins and then averaged again across time series. If no improvement in performance on the validation set was observed for an N-BEATS(-S) extension, we did not produce results for the test set.

First and foremost, by comparing the results of N-BEATS and N-BEATS-S, we can conclude that there certainly exist λ-values that lead to considerable improvements in forecast stability without causing a considerable loss in forecast accuracy for all data subsets considered. Moreover, for the M4-hourly subset, and to a lesser extent for the M4-weekly and M3-monthly subsets, both forecast accuracy and stability improve compared to standard N-BEATS.

The simultaneous improvements in forecast stability and forecast accuracy that are observed for some data subsets indicate that including a forecast instability component in the loss function can be considered an effective time-series-specific regularization mechanism in certain cases. Therefore, we compare the results of N-BEATS-S to N-BEATS with weight decay and N-BEATS with dropout, which are two frequently used regularization techniques for neural networks. For both N-BEATS variants, we only report a test set result if an improvement in validation sMAPE over a standard N-BEATS network was found. For N-BEATS in combination with dropout, in most cases, there were no dropout probabilities that led to improvements in validation sMAPE, and if a non-zero dropout probability was selected, no considerable changes in the test set results were observed. The results for N-BEATS with weight decay indicate that the effect of weight decay on both forecast accuracy and forecast stability is rather limited for the M4 data sets (except for M4-hourly). For the M3 data sets, however, it results in forecasts that are a bit more stable, but it is not as effective as N-BEATS-S at improving forecast stability. The above observations allow us to conclude that, although improving generalization performance in terms of forecast accuracy is not the main objective of our proposed methodology, in most cases, it outperforms traditional regularization techniques from this generalization perspective. Finally, the results for N-BEATS-S with weight decay are reported as well. Also here, the effect of weight decay on both forecast accuracy and forecast stability is limited for the M3-monthly and M4 data sets, whereas for the M3-yearly subset it results in a considerable further improvement in forecast stability albeit at the cost of a decrease in forecast accuracy.

Recall from Section 3.1 that ensembling was used in the original N-BEATS paper to report test set performance. In Oreshkin et al. (2020), an ensemble was constructed based on 180 different N-BEATS networks to report state-of-the-art performance on the M3 and M4 competition data sets. By contrast, our N-BEATS(-S) T ensembles are constructed of ten different networks only, obtained by using five different random initializations for two different lookback window lengths T. Both N-BEATS and N-BEATS-S T ensembles show improvements in both forecast accuracy and forecast stability compared to N-BEATS(-S) with only one lookback window length T for nearly all subsets of the M3 and M4 data sets. By comparing the results of the N-BEATS and N-BEATS-S T ensembles, we can conclude that, although an ensemble with standard N-BEATS networks as base models also tends to result in more stable forecasts, the proposed methodology to improve forecast stability is far more efficient to that end. Finally, note that it is very likely that forecast accuracy (and stability) can be further improved by including more base models in the ensembles with still other values for the lookback window length T (as in Oreshkin et al., 2020). To this end, for the N-BEATS-S T ensemble, adding these extra base models is computationally more expensive than for the N-BEATS T ensemble, since we need to determine a suitable λ-value for each lookback window length T for the former.

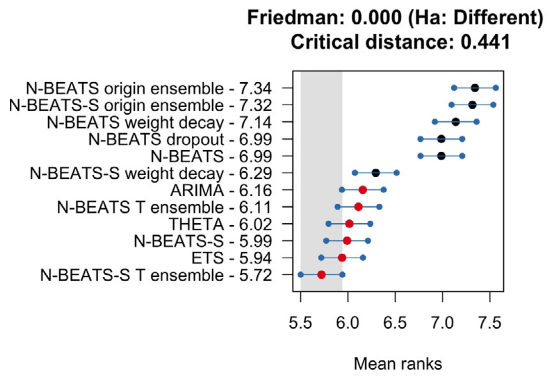
While ensembles are generally used in forecasting with the objective of improving forecast accuracy, in a rolling forecasting exercise, an ensemble forecast to directly reduce forecast instability can be constructed without the need for extra base models. Indeed, by averaging all available forecasts for a specific period and thus reusing forecasts for the considered time period that are made at previous forecasting origins with longer forecast horizons, forecast instability is reduced by construction. We report the results of such *origin ensemble forecasts* for both N-BEATS and N-BEATS-S. The results for both the M3 and M4 data sets effectively show large decreases in forecast instability compared to their respective N-BEATS(-S) counterparts, with the origin ensembles resulting in the two lowest sMAPC values for all data subsets. However, this approach to ensemble construction appears to result in a considerable loss in forecast accuracy for all subsets except M4-hourly. For this data subset, the N-BEATS and N-BEATS-S origin ensembles result in improvements in both stability and accuracy, which is probably a consequence of the large number of input nodes ($7h$ with $h = 24$) for this data subset, resulting in the effect of ensembling on accuracy to weigh more heavily than the effect of one additional data point.

Finally, as benchmarks, we report the results for ETS, ARIMA, and THETA forecasts for the evaluation schemes considered in this paper. For the M3 data sets, in terms of forecast stability, THETA consistently performs better than ETS and ARIMA and also outperforms N-BEATS-S in two out of four cases. Moreover, THETA also results in the most accurate forecasts across all forecasting methods considered in three out of four cases. For the larger M4 data sets, however, the benchmark forecasting methods are no longer competitive with the N-BEATS-S forecasts, as they are generally more stable and accurate. The N-BEATS-S T ensembles give rise to further improvements in forecast stability and forecast accuracy for all data subsets considered. Finally, note that the poor performance of ARIMA for the M4-hourly subset is due to the fact that only non-seasonal ARIMA models are considered for computational reasons.

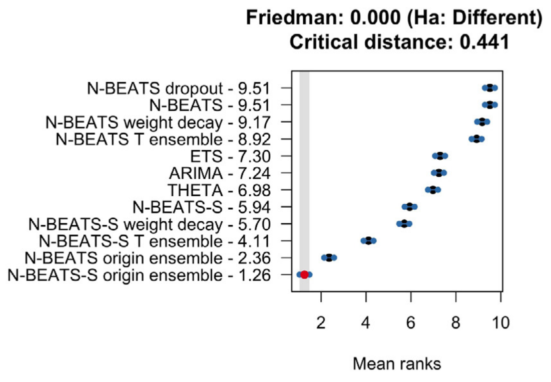
A logical next step is to check whether the reported differences in forecast accuracy and stability in Tables 4 and 5 are also statistically significant. To this end, we visualize the results of multiple comparisons with the best (MCB) tests (Koning, Franses, Hibon, & Stekler, 2005) for the M3-monthly and M4-monthly data sets, in Figs. 4 and 5, respectively. The MCB results for the other subsets are visualized in Appendix A for the M3 data set and in Appendix B for the M4 data set. Essentially, MCB tests are used to determine whether the average (across time series) rank of each method is significantly different from those of other methods. In the MCB plots, if the intervals of two methods do not overlap, this indicates statistically different performances.

The MCB results for the M3-monthly data set, visualized in Fig. 4, clearly confirm that the differences in forecast accuracy and stability between N-BEATS and N-BEATS-S are statistically significant, with N-BEATS-S resulting in more stable and more accurate forecasts. For the other subsets of the M3 data set (see Appendix A), the differences in forecast stability are statistically significant as well. However, in terms of accuracy, N-BEATS and N-BEATS-S perform similarly. We further observe that the origin ensembles perform significantly worse than N-BEATS(-S) (T ensembles) and the benchmark methods in terms of forecast accuracy, whereas they outperform all other methods in terms of stability, with the next best alternatives being N-BEATS-S (T ensemble) and the benchmark methods.

For the M4-monthly data set (see Fig. 5), in terms of forecast accuracy, we can clearly distinguish several groups. The T ensembles statistically outperform all other methods, and are followed by N-BEATS-S with and without weight decay, which outperform the N-BEATS and benchmark methods. In line with the results for the M3 data sets, the origin ensembles perform the worst in terms of accuracy, but outperform all other methods in terms of forecast stability, with the next best alternative being the N-BEATS-S T ensemble. For the other subsets of the M4 data set (see Appendix B), similar conclusions can be drawn, except for the M4-hourly subset, for which the origin ensembles perform well in terms of both forecast accuracy and forecast stability.

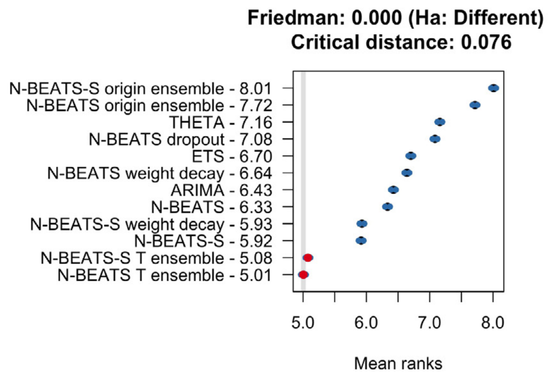


(a) sMAPE ranking

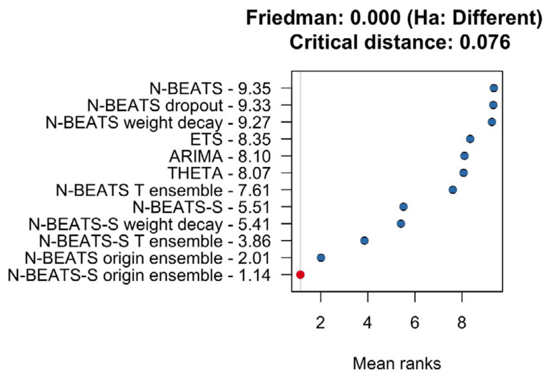


(b) sMAPC ranking

Fig. 4. MCB results for the M3-monthly data set.



(a) sMAPE ranking



(b) sMAPC ranking

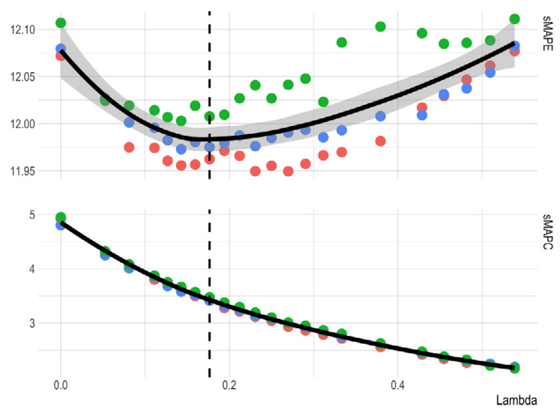
Fig. 5. MCB results for the M4-monthly data set.

4.3. Discussion

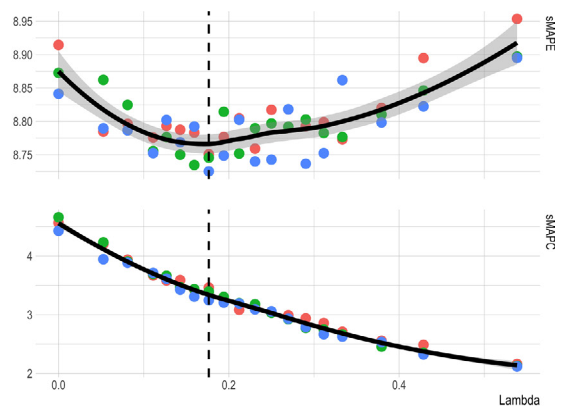
Based on the test results from Tables 4 and 5, we conclude that λ-values exist that yield improvements in forecast stability while maintaining forecast accuracy. Moreover, the results indicate that improvements in both forecast stability and forecast accuracy can be achieved. Therefore, we conjecture that the forecast instability loss component Ω also acts as a regularization mechanism. While Ω indirectly depends on the network parameters $\mathbf{W}$ via the network outputs $\hat{\mathbf{y}}^j_{h|t-1}$ and $\hat{\mathbf{y}}^j_{h|t}$, a regularization term is typically a direct function of the model parameters, i.e., $\Omega(\mathbf{W})$, that imposes a penalty on model complexity to reduce the generalization (test) error and/or to avoid overfitting, possibly at the expense of an increase in training error. The $L1$ and $L2$ norms of the weight vector $\mathbf{W}$ that are added to a linear model in the case of least squares regression, which are known as ridge and lasso regressions (Hastie, Tibshirani, & Friedman, 2011), respectively, are likely the best-known examples of regularization techniques. Adding the (squared) $L2$ norm of the network parameters to the loss function is also a commonly used method in neural networks, known as weight decay in neural network language, whereas in a (local) time series forecasting context, lasso regression is an established method when exogenous information is used in the forecasting process (see, e.g., Sagaert, Aghezzaf, Kourentzes, & Desmet, 2018; Van Belle, Guns, & Verbeke, 2021).

The above regularization techniques all require the selection of a hyperparameter that controls the importance of the regularization term. Similarly, determining a suitable value for λ to control the effect of the forecast instability component on the optimization problem and therefore the model outputs is of crucial importance. This is discussed in the next section.

Validation set performance. The λ-values that generate the test set results are reported in Table 3 and are selected based on minimizing the average sMAPE on the validation set obtained across three different runs with random initializations. Fig. 6 visualizes the validation sMAPE and validation sMAPC for different λ-values and for both the M3-monthly and M4-monthly data sets. Different random seeds that impact network parameter initialization and the generation of training batch sequences are used to build models represented by different colors to enhance the reliability of the results. Note that $\lambda = 0$ represents standard N-BEATS networks.

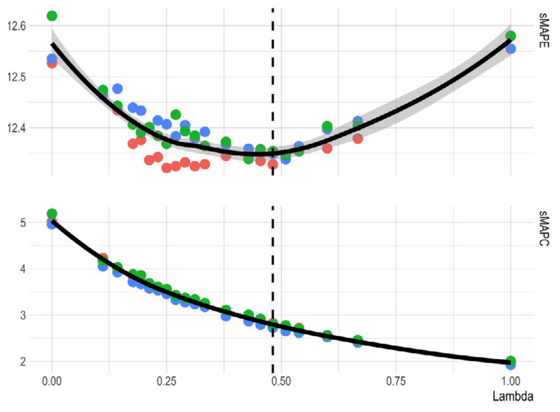


(a) M3-monthly data set

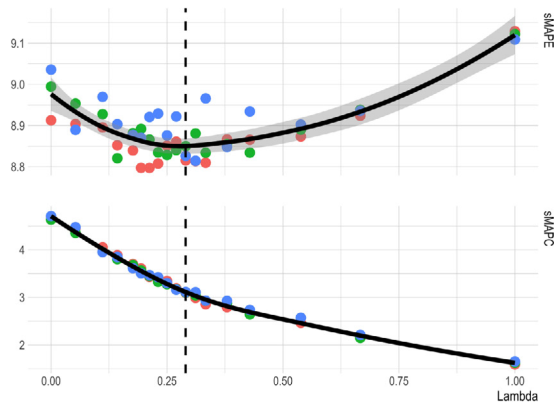


(b) M4-monthly data set

Fig. 6. Effect of λ on validation sMAPE (top panels) and sMAPC (bottom panels) for tuned lookback window lengths T of $6h$ and $4h$. (For the color version of this figure, the reader is referred to the web version of this article.)



(a) M3-monthly data set

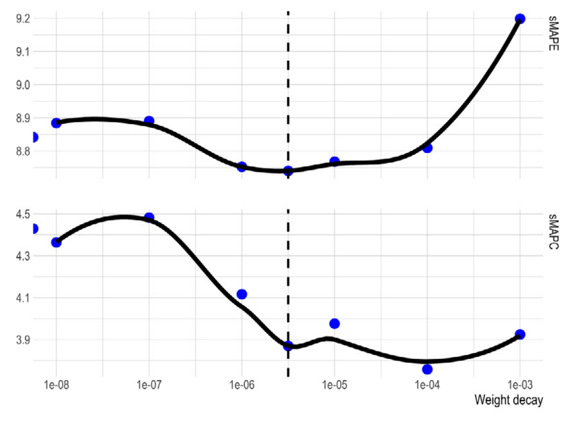


(b) M4-monthly data set

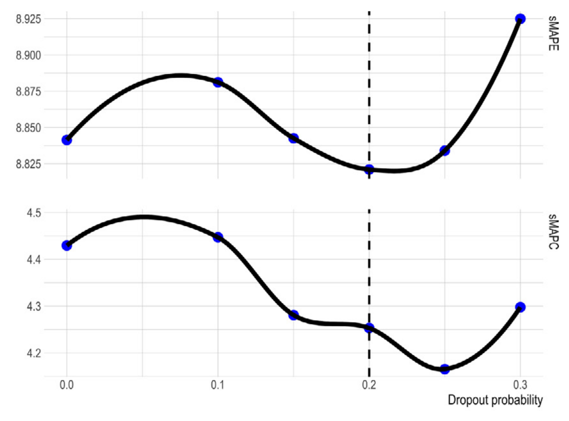
Fig. 7. Effect of λ on validation sMAPE (top panels) and sMAPC (bottom panels) for lookback window lengths $T = 8h$. (For the color version of this figure, the reader is referred to the web version of this article.)

Fig. 6 provides evidence in support of the above conclusions, since the sMAPE and sMAPC values are smaller than those of the standard N-BEATS network for (nearly) all λ-values visualized. For both data subsets, the validation sMAPE first decreases as λ increases and then reaches a minimum value, as indicated in both cases for $\lambda = 0.176$, before rising again for larger λ-values and eventually worsening compared to the validation sMAPE for N-BEATS (i.e., N-BEATS-S with $\lambda = 0$). The evolution of the validation sMAPE with λ-values thus follows the typical pattern that is observed for regularization techniques (see, e.g., Bishop, 2006). Fig. 7 shows similar results for different N-BEATS-S networks with different lookback window lengths T. Here, in contrast to what is observed in Fig. 6, the best values for λ are not equal for both data sets; however, the best λ-values for both the M3-monthly and M4-monthly data sets do increase compared to Fig. 6, which is expected since there are more network parameters for $T = 8h$. Furthermore, note that for the M3-monthly data set, a larger value of λ is better compared to the M4-monthly data set, which is possibly due to the smaller scale of the former (see Table 2).

In Fig. 8, we visualize the effect of weight decay and dropout on the validation sMAPE and sMAPC for the M4-monthly data set and a lookback window length of $T = 4h$ (cf. panel (b) of Fig. 6). On the validation set, the best weight decay results give rise to improvements in forecast accuracy that are quite comparable to that of N-BEATS-S with the selected λ-value. However, as expected, although some improvement in forecast stability is also observed for these weight decay results, the decline in sMAPC is substantially stronger for N-BEATS-S with the selected λ-value (note that different y-axis ranges are



(a) Weight decay



(b) Dropout

Fig. 8. Effect of weight decay and dropout on validation sMAPE (top panels) and sMAPC (bottom panels) for the M4-monthly data set and lookback window length $T = 4h$.

used in Figs. 6(b) and 8(a)). Finally, based on Fig. 8(b), we observe that dropout does not seem to result in considerable improvements to either forecast accuracy or forecast stability.

Learning curves. In this section, we illustrate and highlight the ability of the forecast instability loss component to prevent overfitting via a toy example. To this end, we set up an additional experiment to simulate the availability of only a small data set; we only use the *industry* subset of the M3-monthly data set (334 time series), and we use the same training sample for each time series in each batch; i.e., we use a single forecasting origin per time series. Hyperparameters are similar to those reported in Table 3 for the M3-monthly data set, except for the batch size (256) and the number of iterations (2000). When using deep learning models on small data sets, preventing overfitting is of crucial importance. Fig. 9 visualizes the effect of varying λ on the evolution of training and validation RMSSEs for the above experiment. The left panel of the figure shows that for $\lambda > 0$, training and validation RMSSEs decrease more slowly and that the validation RMSSE converges to a stable level, while the validation RMSSE for a single standard N-BEATS network ($\lambda = 0$) indicates overfitting. The right panel of Fig. 9 clearly visualizes this observation. N-BEATS starts overfitting the training data at approximately iteration 800. By assigning some weight to the forecast instability during optimization, N-BEATS-S with $\lambda = 0.18$ only starts overfitting at iteration 1000, whereas increasing λ to 0.33 resolves the overfitting problem. As such, $\lambda > 0$ ensures that network performance is less sensitive to the number of learning iterations.

5. Conclusions and future research

Forecast instability induced by updating forecasts when new observations become available potentially incurs certain costs. However, these costs are generally considered to be outweighed by the benefits of forecast updating in terms of forecast accuracy improvements. Therefore, point forecast generation and evaluation traditionally focus on forecast accuracy.

In this paper, we proposed an extension to the N-BEATS deep learning architecture for the univariate time series point forecasting problem. Our extension includes forecast (in)stability in the learning phase by introducing a composite loss function that considers both forecast accuracy and stability. The experimental results presented in Section 4 showed that the presented approach results in more stable forecasts without causing a loss in forecast accuracy. As discussed in Section 2.2, in a supply chain planning context, for instance, using stable demand forecasts leads to fewer and smaller revisions to supply plans and thus to lower associated supply chain costs. However, as the experimental study showed, it is possible to improve both forecast accuracy and forecast stability by using the extended loss function (compared to the original N-BEATS architecture). Therefore, we can conclude that the proposed methodology for improving forecast stability can be useful for forecasting applications in general, since it appears to act as a time-series-specific regularization mechanism.

Although we focused on the N-BEATS architecture, in principle, the proposed composite loss function can be used with any deep learning method for the univariate time series point forecasting problem. Moreover, this idea could potentially be generalized to a probabilistic forecasting setting as well, where instability can be captured, for example, by the Kullback–Leibler divergence. We believe that the generalization to a probabilistic setting is an interesting topic for future research, in addition to testing the proposed methodology in combination with the interpretable N-BEATS configuration and other deep learning architectures for the univariate time series point forecasting problem, as well as on other data sets. In this respect, demand forecasting data sets are of special interest given the importance of forecast instability in a supply chain planning context. The fact that no supply-chain-specific data sets were used to test our methodology can

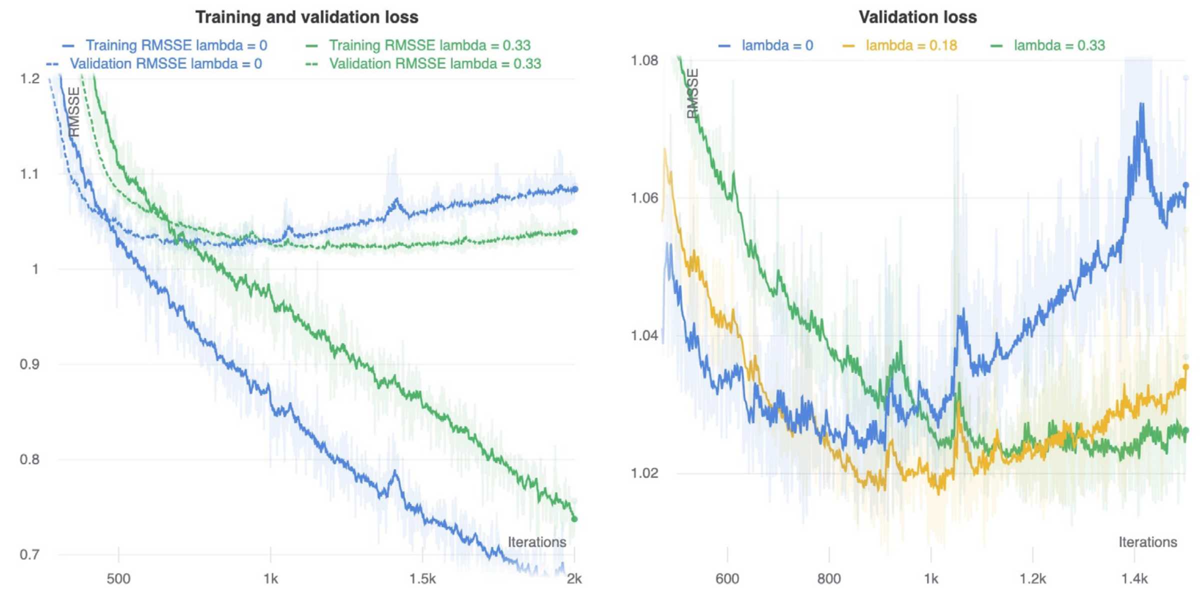


Fig. 9. Effect of λ on the learning curves of training and validation RMSSE for the industry subset of the M3-monthly data set. (For the color version of this figure, the reader is referred to the web version of this article.)

be considered a limitation of our study. Another limitation is that we included fewer networks in our N-BEATS ensembles compared to the original N-BEATS paper (Oreshkin et al., 2020): we only built models for one lookback window length (or two for T ensembles) instead of six different lengths ($T = 2h, \ldots, 7h$); we only used one type of loss function (RMSSE for forecast error and RMSSC for forecast instability) instead of three different loss functions (MAPE, sMAPE, and MASE); and we used five instead of ten different random initializations (for each of the above combinations). Testing the presented methodology in combination with larger ensemble models, therefore, is also an important topic for future work.

Other possibilities for future research include direct extensions of the proposed composite loss function and/or modifications to the training scheme that is used. For instance, by considering multiple lagged lookback windows (shifted further back in time), we can take into account *multi-period forecast instability* in the optimization problem as well. For example, in addition to $\Omega(\hat{\mathbf{y}}_{h|t-1}, \hat{\mathbf{y}}_{h|t})$, we could also include $\Omega(\hat{\mathbf{y}}_{h|t-2}, \hat{\mathbf{y}}_{h|t})$. In this regard, it is reasonable to downweight the importance of multi-period forecast instability for increasing gaps between the respective forecasting origins. As such, this possible extension also raises new questions. In regard to the training scheme that is used, in the exposition of the proposed extension to N-BEATS for improving forecast stability in Section 3.2, λ is presented as a static hyperparameter that controls the extent to which the forecasts produced by the network are influenced by the forecast instability component. However, since we rely on gradient descent to optimize the neural networks, adaptive weighting schemes could alternatively be used to dynamically balance the two loss components that constitute the composite loss function, i.e., forecast error and forecast instability.

A final direction for future research concerns an investigation of the potential complementarity of the proposed forecast instability loss component and other *standard* regularization techniques. This would be an interesting topic, since our experimental results indicate that extra performance gains can be achieved in terms of both forecast accuracy and forecast stability by using an ensemble of N-BEATS-S networks with only two different lookback window lengths T.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

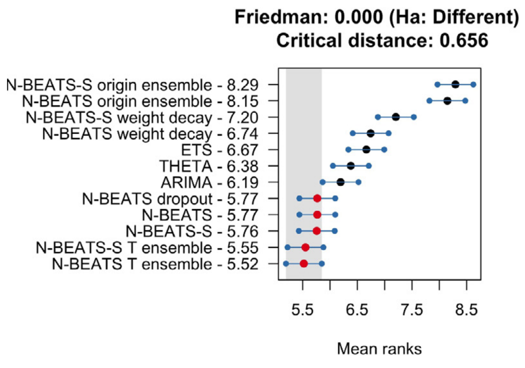
This research is funded by Agentschap Innoveren & Ondernemen Vlaanderen (project HBC.2016.0589) and OMP. We would also like to thank the anonymous reviewers for their insightful comments which helped us to improve the paper.

Appendix A. MCB results for the other subsets of the M3 data set

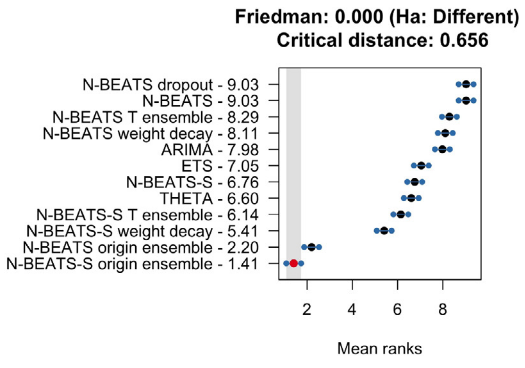
See Figs. A.1–A.3.

Appendix B. MCB results for the other subsets of the M4 data set

See Figs. B.1–B.5.

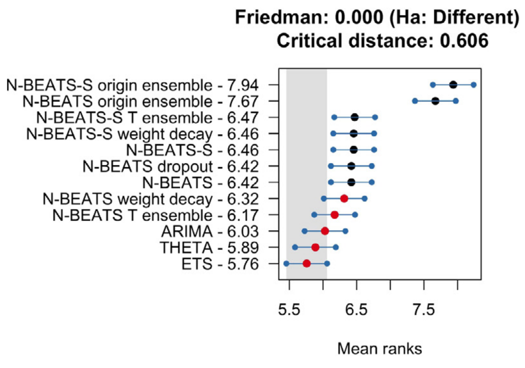


(a) sMAPE ranking

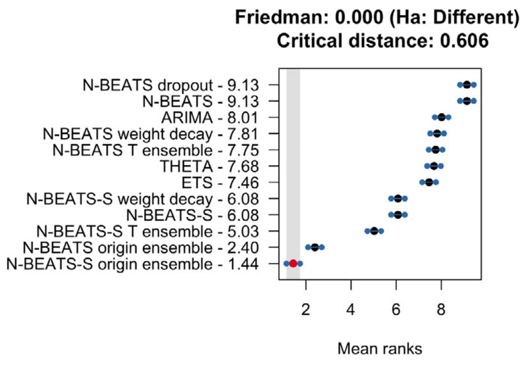


(b) sMAPC ranking

Fig. A.1. MCB results for the M3-yearly data set.

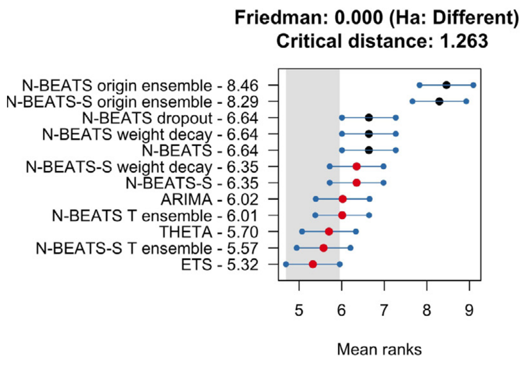


(a) sMAPE ranking

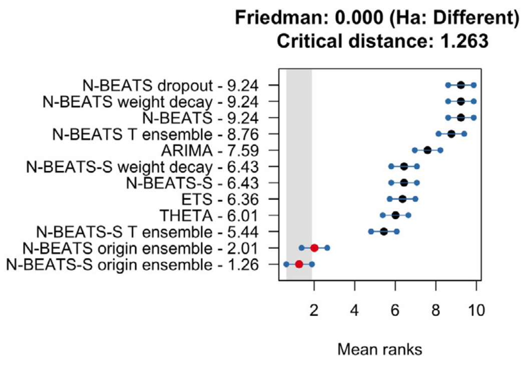


(b) sMAPC ranking

Fig. A.2. MCB results for the M3-quarterly data set.

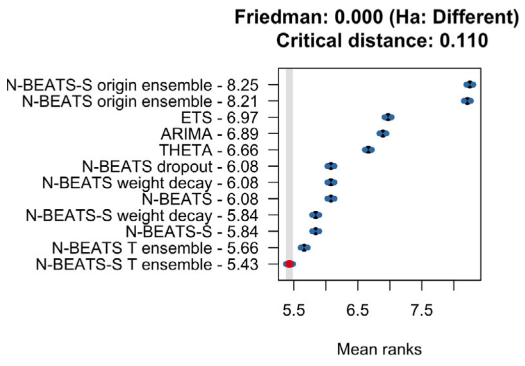


(a) sMAPE ranking

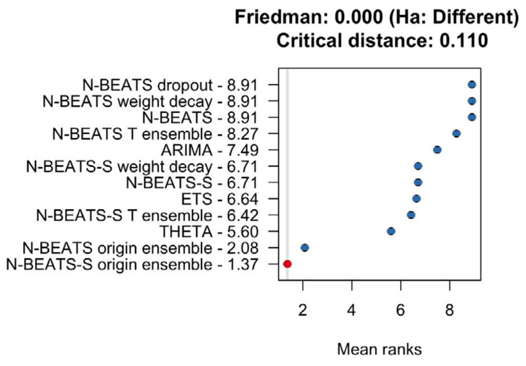


(b) sMAPC ranking

Fig. A.3. MCB results for the M3-other data set.

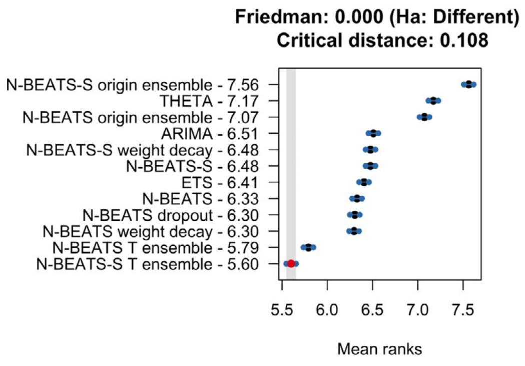


(a) sMAPE ranking

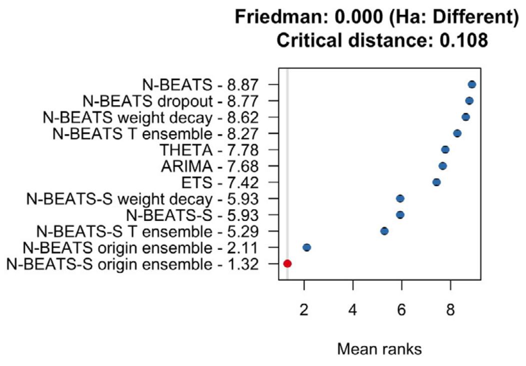


(b) sMAPC ranking

Fig. B.1. MCB results for the M4-yearly data set.

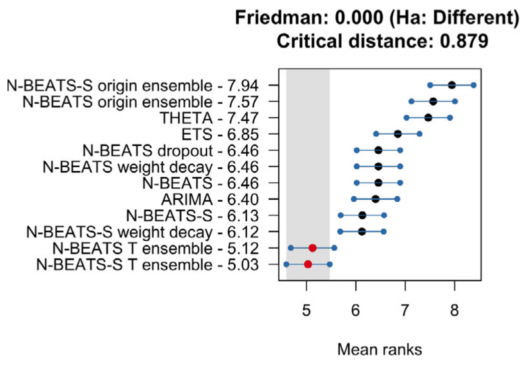


(a) sMAPE ranking

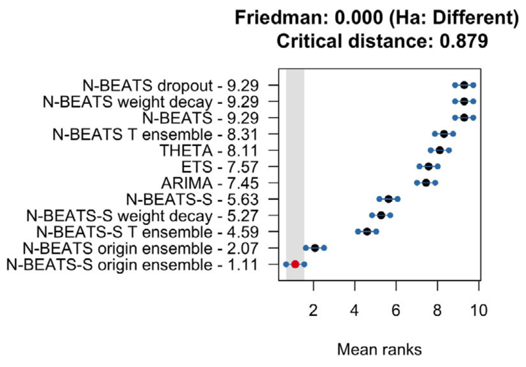


(b) sMAPC ranking

Fig. B.2. MCB results for the M4-quarterly data set.

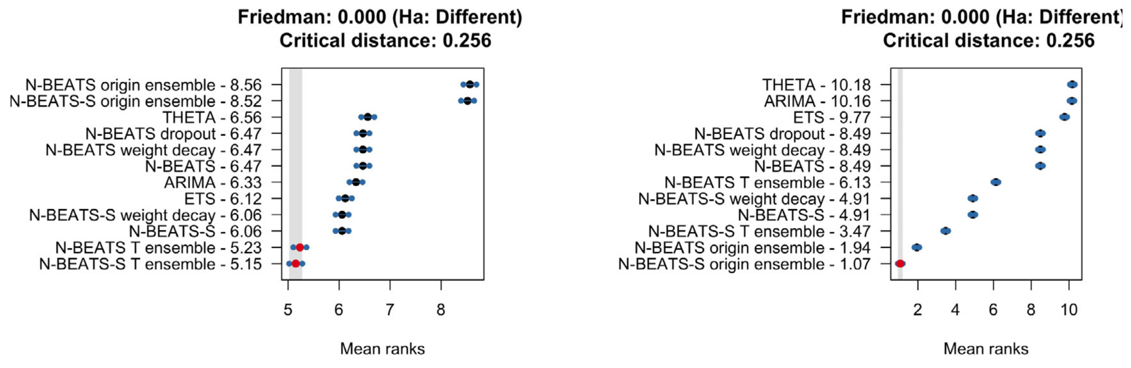


(a) sMAPE ranking



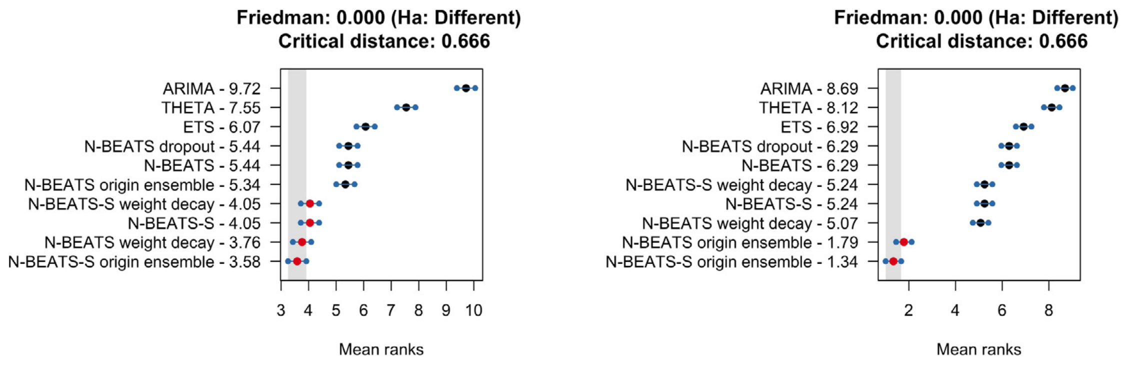
(b) sMAPC ranking

Fig. B.3. MCB results for the M4-weekly data set.



(a) sMAPE ranking (b) sMAPC ranking

Fig. B.4. MCB results for the M4-daily data set.



(a) sMAPE ranking (b) sMAPC ranking

Fig. B.5. MCB results for the M4-hourly data set.

References

Altavilla, C., & Ciccarelli, M. (2007). *Information combination and forecast (st)ability: Evidence from vintages of time-series data*: *Technical Report 846*, European Central Bank (ECB), https://ideas.repec.org/p/ecb/ecbwps/2007846.html.

Assimakopoulos, V., & Nikolopoulos, K. (2000). The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, *16*(4), 521–530, https://www.sciencedirect.com/science/article/pii/S0169207000000662.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer, https://link.springer.com/book/9780387310732.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140, https://link.springer.com/article/10.1007/BF00058655.

Chatfield, C. (1996). Model uncertainty and forecast accuracy. *Journal of Forecasting*, *15*(7), 495–508, https://doi.org/10.1002/(SICI)1099-131X(199612)15:7<495::AID-FOR640>3.0.CO;2-O.

Hastie, T., Tibshirani, R., & Friedman, J. (2011). *The Elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). NY: Springer, https://link.springer.com/book/10.1007/978-0-387-84858-7.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., et al. (2020). Forecast: Forecasting functions for time series and linear models. https://pkg.robjhyndman.com/forecast/, R package version 8.13.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, *26*(3), 1–22, https://www.jstatsoft.org/article/view/v027i03.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688, https://www.sciencedirect.com/science/article/pii/S0169207006000239.

Hyndman, R. J., Koehler, A. B., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach*. Berlin: Springer-Verlag, https://link.springer.com/book/10.1007/978-3-540-71918-2.

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, *18*(3), 439–454, https://www.sciencedirect.com/science/article/pii/S0169207001001108.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., et al. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, *36*(1), 167–177, https://www.sciencedirect.com/science/article/pii/S0169207019301529.

Januschowski, T., Kolassa, S., et al. (2019). A classification of business forecasting problems. *Foresight: The International Journal of Applied Forecasting*, (52), 36–43, https://ideas.repec.org/a/for/ijafaa/y2019i52p36-43.html.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kolassa, S. (2011). Combining exponential smoothing forecasts using akaike weights. *International Journal of Forecasting*, *27*(2), 238–251, https://www.sciencedirect.com/science/article/pii/S0169207010001032.

Koning, A. J., Franses, P. H., Hibon, M., & Stekler, H. O. (2005). The M3 competition: statistical tests of the results. *International Journal of Forecasting*, *21*(3), 397–409, https://www.sciencedirect.com/science/article/pii/S0169207004000810.

Li, Q., & Disney, S. M. (2017). Revisiting rescheduling: MRP nervousness and the bullwhip effect. *International Journal of Production Research*, *55*(7), 1992–2012, https://doi.org/10.1080/00207543.2016.1261196.

Makridakis, S., & Hibon, M. (2000). The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476, https://www.sciencedirect.com/science/article/pii/S0169207000000571.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4-competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74, https://www.sciencedirect.com/science/article/pii/S0169207019301128.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1ecqn4YwB.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc., https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Sagaert, Y. R., Aghezzaf, E.-H., Kourentzes, N., & Desmet, B. (2018). Tactical sales forecasting using a very large set of macroeconomic indicators. *European Journal of Operational Research*, *264*(2), 558–569, https://www.sciencedirect.com/science/article/pii/S0377221717305957.

Schuster, N., Ehm, H., Hottenrott, A., & Lauer, T. (2017). Bridging short and mid-term demand forecasting in the semiconductor industry. In *2017 Winter simulation conference* (pp. 3658–3669). IEEE, https://ieeexplore.ieee.org/document/8248078.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85, https://www.sciencedirect.com/science/article/pii/S0169207019301153.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958, http://jmlr.org/papers/v15/srivastava14a.html.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, *16*(4), 437–450, https://www.sciencedirect.com/science/article/pii/S0169207000000650.

Tunc, H., Kilic, O. A., Tarim, S. A., & Eksioglu, B. (2013). A simple approach for assessing the cost of system nervousness. *International Journal of Production Economics*, *141*(2), 619–625, https://www.sciencedirect.com/science/article/pii/S0925527312004185.

Van Belle, J., Guns, T., & Verbeke, W. (2021). Using shared sell-through data to forecast wholesaler demand in multi-echelon supply chains. *European Journal of Operational Research*, *288*(2), 466–479, https://www.sciencedirect.com/science/article/pii/S0377221720305208.

Yuan, Z., & Yang, Y. (2005). Combining linear regression models: When and how? *Journal of the American Statistical Association*, *100*(472), 1202–1214, https://doi.org/10.1198/016214505000000088.

Zou, H., & Yang, Y. (2004). Combining time series models for forecasting. *International Journal of Forecasting*, *20*(1), 69–84, https://www.sciencedirect.com/science/article/pii/S0169207003000049.