Contents lists available at ScienceDirect

International Journal of Forecasting

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

Hierarchical forecasting with a top-down alignment of independent-level forecasts

Matthias Anderer[a], Feng Li[b,*]

[a] Matthias Anderer GmbH, Holzkirchen, Germany
[b] School of Statistics and Mathematics, Central University of Finance and Economics, Beijing 102206, China

ARTICLE INFO

Keywords:
M5 competition
Forecasting reconciliation
Hierarchical forecasting
Hierarchical alignment
Deep learning forecasting

ABSTRACT

Hierarchical forecasting with intermittent time series is a challenge in both research and empirical studies. Extensive research focuses on improving the accuracy of each hierarchy, especially the intermittent time series at bottom levels. Then, hierarchical reconciliation can be used to improve the overall performance further. In this paper, we present a hierarchical-forecasting-with-alignment approach that treats the bottom-level forecasts as mutable to ensure higher forecasting accuracy on the upper levels of the hierarchy. We employ a pure deep learning forecasting approach, N-BEATS, for continuous time series at the top levels, and a widely used tree-based algorithm, LightGBM, for intermittent time series at the bottom level. The hierarchical-forecasting-with-alignment approach is a simple yet effective variant of the bottom-up method, accounting for biases that are difficult to observe at the bottom level. It allows suboptimal forecasts at the lower level to retain a higher overall performance. The approach in this empirical study was developed by the first author during the M5 Accuracy competition, ranking second place. The method is also business orientated and can be used to facilitate strategic business planning.

1. Introduction

With the rise of business-oriented time series data collection across industries, improving forecasting performance has become vital for business success. Among the many forecasting techniques, conventional statistical forecasting methods are dominant in the forecasting domain; for a survey in this area, see Hyndman and Athanasopoulos (2018), and for an encyclopedic review of forecasting theory and applications, see Petropoulos et al. (2021)). Machine learning algorithms are widely used in time series forecasting, notably long short-term memory (LSTM) neural networks and their variations, deep neural architectures such as N-BEATS (Oreshkin et al., 2019), and the temporal fusion transformer (Lim et al., 2019). The top-ranking method at the M4 competition (Smyl, 2020) suggested that hybrid models that combine statistical approaches and machine learning methods could further improve forecasting accuracy.

Although statistical approaches tend to be more interpretable, a cost-efficient forecast usually requires expert information in, e.g., feature engineering and model specification. Those approaches work well with individual time series, but it is hard to scale them up with sizeable related time series sets. On the contrary, industrial machine learning platforms with pipelines provided for global forecasting solutions do not necessarily achieve the best optimality without extensive tuning. Still, they are easy to scale up for massive amounts of time series data. It is thus appealing for industrial forecasters to adopt machine learning approaches.

Intermittent time series, which contain many zeros in the data, are a typical case in demand forecasting. Such time series are difficult to forecast because it is

* Corresponding author.
E-mail addresses: matthias@matthiasanderer.de (M. Anderer), feng.li@cufe.edu.cn (F. Li).

https://doi.org/10.1016/j.ijforecast.2021.12.015

toilsome to make realistic assumptions across all the time series horizons. Dating back to the 1970s, researchers observed that conventional time series methods, such as exponential smoothing, did not perform well for forecasting intermittent-demand time series (Croston, 1972). A straightforward approach, known as Croston's method, was proposed by splitting the time series into the non-zero component and stochastic component. The two parts were then forecasted separately, and the final forecast was factorized by the two individual forecasts. See, e.g., Croston (1972), Rao (1973), Shenstone and Hyndman (2005) for related studies. Although such a method has proven helpful, it often produces forecast bias (Levé & Segerstedt, 2004). Many variants were proposed to improve the accuracy (Kourentzes & Athanasopoulos, 2021; Shenstone & Hyndman, 2005) or reduce the bias (Nikolopoulos et al., 2011; Syntetos & Boylan, 2005) of Croston's method.

Attracted by the nonparametric and assumption-free nature of neural networks, researchers shifted their focus to neural networks (Canera Turkmen et al., 2020), and directly constructed neural networks for time series with particular patterns, most notably as short intermittent time series (Kourentzes, 2013) and long lumpy time series (Gutierrez et al., 2008). One widely known approach is DeepAR—an architecture based on autoregressive recurrent networks (Salinas et al., 2020).

Another challenge in the supply chain or online sales industry is simultaneously forecasting an extensive collection of time series. Such time series collection usually involves a hierarchical structure from different cross-sectional levels. For example, the dataset in the M5 competition (Makridakis et al., 2021) consisted of 3049 products sold across ten stores located in three states in the U.S. The products were further classified into three product categories and seven product departments. Notably, information from different cross-sectional levels may be crucial for improving overall forecasting performance. Although practitioners have found that incorporating cross-sectional information improves the overall forecasting performance, a challenge exists that the forecasts on each hierarchy should be coherent with the aggregation structure of the collection of time series. A hierarchical forecasting method could utilize the structural information to improve the overall forecasting performance and retain forecast coherence. See Hyndman and Athanasopoulos (2018) for an introduction to hierarchical forecasting.

A common phenomenon of hierarchical time series sets in the sales industry is that the lowest level of the hierarchy exhibits a strong intermittent pattern, and the upper hierarchy levels contain forecastable components such as the trend or seasonality aggregated by the bottom level. Early research shows that spatial aggregation for intermittent time series improves the overall forecast accuracy (Zufferey et al., 2016). Similar results were found with temporally aggregated intermittent time series in Nikolopoulos et al. (2011) and Kourentzes and Athanasopoulos (2021).

To this end, we consider a typical business planning scenario. Under certain circumstances, managers at headquarters or investors may not be interested in the forecasts of a particular product in a specific store. Rather, they may be more concerned with the state level's forecasts for all products, which affect overall revenue. By further incorporating the product information as explanatory variables into the bottom-level models, we argue that decision-makers or business planners may benefit by further improving forecasting performance, especially at the top levels, from hierarchical forecasting with intermittent time series, than by merely using individual forecasting models for the bottom-level intermittent data.

In this paper, we describe and analyze an approach proposed by the first author, who utilized standard machine learning toolchains with a careful forecasting alignment scheme that achieved second place at the M5 Accuracy competition. Unlike the forecasting reconciliation approach, which designs optimal proportions for the aggregations to guarantee coherence across multi-hierarchy forecasts, the proposed hierarchical-forecasting-with-alignment approach adjusts the sum of the forecasts of the bottom level based on those aggregated at the top levels by selecting a tuning parameter at the bottom-level model so that the forecasts produced across multiple forecasting horizons are aligned. Our analysis finds it particularly useful when there is difficulty finding an optimal reconciliation, as suggested in, e.g., Wickramasuriya et al. (2019). A vital feature of this approach is that it improves the overall forecast accuracy by allowing some suboptimal forecasts at the lower level while retaining high forecasting accuracy at the top levels.

We organize the remainder of the paper as follows. Section 2 presents preliminaries for the models used in our paper. The methodology is described in Section 3. Section 4 describes the implementation details for the M5 competition. An ex post analysis is provided in Section 5, and Section 6 concludes the paper.

2. Preliminaries

This section briefly describes two machine learning algorithms, i.e., N-BEATS (Oreshkin et al., 2019) and LightGBM (Ke et al., 2017), that are used for the proposed hierarchical-forecasting-with-alignment framework. Both methods are robust and widely used in the machine learning community.

N-BEATS (Oreshkin et al., 2019) is a pure deep learning approach with a deep neural architecture based on backward and forward residual links. It utilizes a very deep stack of fully connected layers that consist of 30 stacks of depth 5 with a total depth of 150 layers. N-BEATS constructs a doubly residual stacking architecture with two residual branches. One residual branch runs over the backcast prediction of each layer, and the other runs over the forecast branch of each layer. This architecture improves the interoperability of deep learning models by allowing for trend and seasonality decomposition. A desired feature for practitioners is that the N-BEATS model does not require massive data sources, complex data transformation, or feature engineering, making the model extremely easy to use and deploy in the industry.

The performance of the N-BEATS model relies on an ensembling step, a forecasting combination technique that combines the forecasts from different models. The final forecasts provided by N-BEATS are based on multi-scale aggregation with the median of bagging selected models. The model pool for ensembling consists of N-BEATS models fitted on one or several of the sMAPE, MASE, and MAPE metrics and a selection of different window lengths, e.g., $2 \times h$, $3 \times h, \ldots, 7 \times h$, where h is the forecast horizon.

The N-BEATS model is robust and has been already been implemented on industry-level platforms. We use the `GluonTS` implementation (Alexandrov et al., 2020) in this analysis. The notable difference between the `GluonTS` implementation and the original N-BEATS pertains to splitting the training and forecasting series. N-BEATS picks a random forecast point from the historical range of length L_h for each selected time series, immediately preceding the last point in the training part of the time series. This method requires further tuning of the hyperparameter L_h. However, `GluonTS` does not use the L_h parameter and cuts time windows by randomly selecting a time series and a starting point on that time series, which reduces the computation burden.

LightGBM (Ke et al., 2017) is a widely used machine learning algorithm based on gradient boosting decision trees. The decision tree algorithm was developed for machine learning tasks such as regression and classification. It partitions the input variables into tree structures, and the final prediction is decided based on the tree of input variables. The decision tree is a weak learner, due to its simple decision strategy. Gradient boosting algorithms are then used to ensemble the predictions in a step-wise fashion to improve the prediction performance. LightGBM further enhances the efficiency and scalability of gradient boosting libraries such as XGBoost (Chen & Guestrin, 2016) for large datasets with high-dimensional features.

3. Methodology

3.1. Improved N-BEATS ensembles for upper levels

Hierarchical forecasting aims to build suitable forecasting models at different levels and to prevent overfitting at all levels using the hierarchical structure. We first train the N-BEATS model using data from the top five levels. Although the data from the top five levels are used to train the N-BEATS model, only the forecasts at the top level are employed for the hierarchical alignment described in Section 3.3. The other four levels serve as a cross-check.

During the M5 competition, the first author experimented with N-BEATS ensembles for the top-level time series during the validation timeframe with different epochs and found that the N-BEATS model started to overfit after approximately 12 epochs. For that reason, N-BEATS ensembles for the top-level forecast with 10 epochs were chosen for the final evaluation models. A comparison with other single forecasting models on the top level was not made due to the length of the competition time and the limited computing resources available on the Kaggle platform. But according to the reported score by the competitors[1] on the Kaggle platform, single models like LightGBM at the top levels do not achieve better accuracy.

The stochastic gradient descent (SGD) algorithm used in the N-BEATS model has weak learning stability. The learning accuracy is very sensitive to the learning rate, and the convergence rate is slow. The *lookahead* optimizer (Zhang et al., 2019) is adopted to the N-BEATS model trainer to improve the training accuracy and learning stability. Unlike the default stochastic searching scheme used in the SGD, the *lookahead* algorithm chooses a parameter search direction by looking ahead at the sequence of weights generated by the standard optimizer provided by the `GluonTS` package.

3.2. Bias-adjusted LightGBM model for the bottom level

At the bottom level, we use a bias-adjusted LightGBM to model the intermittent time series. The root mean squared error (RMSE) loss is used in LightGBM as follows:

$$RMSE = \sqrt{\frac{1}{h}\sum_{t=n+1}^{n+h}(Y_t - \hat{Y}_t)^2}.$$

where Y_t is the true value, $\hat{Y}_t$ is the forecast, h is the forecasting horizon, and the residual is defined as $e_t = Y_t - \hat{Y}_t$. We further use a customized gradient for the RMSE loss as follows:

$$\text{gradient} = \begin{cases} -2e_t & e_t < 0, \\ -2\lambda e_t & e_t \geq 0 \end{cases} \tag{1}$$

where $\lambda > 0$ is a tuning parameter, called the loss multiplier, to allow for an asymmetric loss. When $e_t \geq 0$ (i.e., when the forecast is lower than the true value), the loss multiplier magnifies ($\lambda > 1$) or minifies ($0 < \lambda < 1$) the gradient of the loss function during the learning process. When $\lambda = 1$, the customized gradient reduces to the conventional symmetric gradient. To this end, the loss multiplier controls the bias during the training process to achieve a bias–variance tradeoff effect for the bottom-level model. The customized gradient with the tuning parameter is particularly useful for the final hierarchical alignment step described in Section 3.3. The corresponding Hessian is

$$\text{Hessian} = \begin{cases} 2 & e_t < 0, \\ 2\lambda & e_t \geq 0. \end{cases}$$

We independently train separate models for the time series data at each store for parallelization simplicity on the Kaggle computing platform. It is worth mentioning that no rolling or lagged demand features were used in the LightGBM training, in contrast to conventional statistical forecasting approaches. The motivation is to focus on the intermittent characteristics at the bottom level. We argue that intermittent demand can be significantly influenced by factors like prices, time of the year, or special events, rather than the historical demand information. This approach with non-time series features consistently achieves more stable forecast results. For the final forecasts, five different multipliers around the optimal loss multiplier were used to build five independent forecast

[1] Available online at https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/134712.

models. The final forecasts at the bottom level are based on the mean ensemble of the five models. See Section 4 for the implementation details.

During the M5 competition, the first author also investigated the DeepAR (Salinas et al., 2020) approach for modeling the intermittent time series. Nonetheless, the experiments found it difficult to achieve better results than LightGBM models or to obtain optimal reconciliation results compared with the minimum trace (MinT) optimal reconciliation approach proposed in Wickramasuriya et al. (2019).

3.3. Aligning top-level forecasts and aggregated bottom-level forecasts

Given the independent forecasting results for the top level from stable N-BEATS forecasting models with 10 epochs, the final step is to find the optimal bottom-level forecasts produced by the LightGBM model. We tune the loss multiplier (λ) introduced for the LightGBM model in Section 3.2 so that the RMSE between N-BEATS forecasts and aggregated bottom-level forecasts reaches a minimum. This step is called the hierarchical alignment, and the corresponding alignment metric is defined as follows:

$$\arg\min_{\lambda}\left\{\sqrt{\frac{1}{h}\sum_{t=n+1}^{n+h}\left(\hat{Y}_t^{(\mathrm{top})}-Agg(\hat{Y}_t^{(\mathrm{bottom})}(\lambda))\right)^2}\right\}, \qquad (2)$$

where $Agg(\cdot)$ is the aggregating method used at the bottom level. The mean aggregation function was used throughout the M5 competition.

The hierarchical-forecasting-with-alignment method is simple and applicable to any model appropriate to the bottom level. The loss multiplier λ in Eq. (2) is selected so that the forecasts produced across the entire forecasting horizon are aligned. This means that the objective function for the alignment depends on the forecasting horizon. During the M5 competition, a manual grid search method was used to find the optimal loss multiplier for the LightGBM model at the bottom level, which can easily be automated in an industrial setting. This hierarchical alignment method only requires running the model for the top level once. The optimal loss multiplier can be obtained during the optimization stage of the bottom-level model by examining the alignment metric in Eq. (2). Note that in the evaluation timeframe during the M5 competition, Eq. (2) was used to determine the optimal multiplier values because the actual future values were unknown. We provide an ex post analysis in Section 5 using the WRMSSE metric (Eq. (4)) now that all data and results have been disclosed following the competition.

It is worth mentioning that there are alternative ways to achieve hierarchical alignment. In principle, this hierarchical alignment method should work for aligning all upper levels with the bottom level. The reason that we only align with the top level is twofold. First, the scoring metric in the M5 competition gives higher weights on upper levels because they contain fewer series than the bottom level. Second, the computational cost on the top level is much lower than aligning with several levels.

4. Implementation details

The hierarchical-forecasting-with-alignment approach was successfully applied to the M5 competition dataset, consisting of a hierarchical structure of daily sales data of 42,840 series spanning 1941 days. Table 1 depicts the hierarchical structure of the data and the number of series per aggregation level. The table shows that the upper levels contain far fewer series than the lower level.

The root mean squared scaled error (RMSSE), which is a variant of the well-known mean absolute scaled error (MASE) (Hyndman & Koehler, 2006), was used to calculate the out-of-sample forecasting error for each series:

$$RMSSE = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}(Y_t-\hat{Y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(Y_t-Y_{t-1})^2}}. \qquad (3)$$

After estimating the RMSSE for all 42,840 time series of the competition, the weighted RMSSE (WRMSSE) was used by the organizer for the overall accuracy comparison defined in Eq. (4)

$$WRMSSE = \sum_{i=1}^{42,840}\omega_i \times RMSSE_i \qquad (4)$$

where ω_i is the weight of the series based on actual dollar sales of the product. It is worth mentioning that the actual future values $(Y_{n+1},\ldots,Y_{n+h})$ were not available during the validation frame in the M5 competition.

Interestingly, one may notice that the RMSSE from hierarchical levels of view are equally weighted, which indicates that the overall accuracy in WRMSSE is primarily affected by upper-level forecasts. Because all hierarchical levels are equally weighted with the M5 accuracy metric, WRMSSE, it is easier to forecast continuous time series at upper levels than to forecast massive intermittent time series at the bottom level. Based on the above concern, the hierarchical alignment scheme builds the forecasting strategy by focusing on the forecasting accuracy at the top-level forecasts. It aligns them with the bottom-level forecasts.

Table 2 shows the time series features used for the LightGBM model at the bottom level. The feature matrix includes all available categorical features, like store_id and category_id. Furthermore, the time since the first sale of an item, the price, and the derived price features (price_min, price_max, etc.) are also included. Events and SNAP data, and time features such as the day, week, and month, are also used in the model.

Tables 3 and 4 document the parameter settings for the N-BEATS model and LightGBM model, respectively. The baseline features for the bottom-level LightGBM model and hyperparameters were taken from a public Kaggle user notebook provided by Konstantin Yakovlev (https://www.kaggle.com/kyakovlev/m5-simple-fe). The feature matrix does not include any time rolling or lagged features and we find that the non-time series features are of great importance for intermittent time series forecasting with the LightGBM model. There are two hyperparameters: (i) the number of epochs for the N-BEATS model on the top level, and (ii) the optimal loss multiplier for the top-down alignment. The approach requires

Table 1
Overview of series per level and contribution to error metric for that level. Note that all hierarchical levels were equally weighted in the M5 competition.

Hierarchy level	Description	Number of series
1	All products, all stores, all states	1
2	All products by state	3
3	All products by store	10
4	All products by category	3
5	All products by department	7
6	Unit sales of all products, aggregated for each state and category	9
7	Unit sales of all products, aggregated for each state and department	21
8	Unit sales of all products, aggregated for each store and category	30
9	Unit sales of all products, aggregated for each store and department	70
10	Unit sales of product *x*, aggregated for all stores/states	3,049
11	Unit sales of product *x*, aggregated for each state	9,147
12	Unit sales of product *x*, aggregated for each store	30,490
Total		42,840

Table 2
Features for intermittent time series data at the bottom level used in LightGBM.

Feature	Description
item_id	3049 unique identifiers of items.
dept_id	16,942 unique identifiers of departments.
cat_id	5652 unique identifiers of categories, e.g., foods, household, hobbies.
sell_price	Price of item in store on a given date.
event_type	108 categorical events, e.g. sporting, cultural, religious.
event_name	157 event names for event_type, e.g. Super Bowl, Valentine's Day, Presidents' Day.
event_name_2	Name of event feature as given in competition data.
event_type_2	Type of event feature as given in competition data.
snap_CA	Binary indicator for SNAP information in CA.
snap_TX	Binary indicator for SNAP information in TX.
snap_WI	Binary indicator for SNAP information in WI.
release	Release week of item in store.
price_max	Maximum price for item in store in the training data.
price_min	Minimum price for item in store in the training data.
price_std	Standard deviation of price for item in store in the training data.
price_mean	Mean of price for item in store in training data.
price_norm	Normalized sell price. divided by the price_max.
price_nunique	Number of unique prices for item in store.
item_nunique	Number of unique items for a given price in store.
price_diff_w	Weekly price changes for items in store.
price_diff_m	Price changes of item in store compared to its monthly mean.
price_diff_y	Price changes of item in store compared to its yearly mean.
tm_d	Day of month.
tm_w	Week in year.
tm_m	Month in year.
tm_y	Year index in the training data.
tm_wm	Week in month.
tm_dw	Day of week.
tm_w_end	Weekend indicator.

Table 3
Parameter settings for N-BEATS ensembles using `GluonTS` (Alexandrov et al., 2020).

Parameter	Description	Value
`num_stacks`	The number of stacks the network should contain.	30
`widths`	Widths of the fully connected layers with ReLu activation in the blocks.	512
`meta_prediction_length`	Forecast horizon h.	28
`meta_bagging_size`	The number of models that share the parameter combination. Each of these models gets a random initialization.	3
`meta_context_length`	The number of time units that condition the predictions.	$h \times \{3, 5, 7\}$
`meta_loss_function`	The loss function (metric) to use for training the models.	`sMAPE`
`learning_rate`	Learning rate for each boosting round.	0.0006
`epochs`	The number of epochs used for the optimization algorithm.	$\{10, 12\}$
`num_batches_per_epoch`	The number of batches in each epoch for the optimization algorithm.	1000
`batch_size`	The batch size used in the optimization.	16

minimal effort for parameter tuning. We use a simple grid search on the validation window to find the optimal hyperparameters.

Finally, in the evaluation phase of the M5 competition, the optimal multiplier value used on the evaluation timeframe was 0.95, based on a manual grid search in the

Table 4
Parameter settings for the LightGBM model (Ke et al., 2017).

Parameter	Description	Value
`objective`	The objective function to maximize with an optimization algorithm	`custom`
`metric`	Metric(s) to be evaluated on the evaluation set	`rmse`
`learning_rate`	Learning rate for each boosting round.	0.2
`lambda_l1`	L_1 norm penalty to prevent overfitting	0.5
`lambda_l2`	L_2 norm penalty to prevent overfitting	0.5
`bagging_freq`	Frequency for bagging at every k iterations	1
`bagging_fraction`	The proportion of randomly selected data for the next k iterations.	0.85
`colsample_bytree`	The proportion for randomly selecting a subset of features on each tree.	0.85
`colsample_bynode`	Proportion for randomly selecting a subset of features on each tree node.	0.85

space of $(0, 2]$. To produce the final output, we further use the mean ensemble to obtain the results based on the five closest multipliers [0.90, 0.93, 0.95, 0.97, 0.99] in the grid around the optimal value of 0.95. This ensemble method reduces the high variance effect on the bottom-level forecasts when we apply LightGBM with the features used in Table 2. Although decision-makers focus on the accuracy of top levels, the low-variance forecasts on the bottom level also improve the overall performance. The extra ensemble step of the five models around the best λ mitigates this high variance while preserving the low bias. Fig. 1 visualizes the N-BEATS forecasts and the aggregated forecasts on the top five levels during the evaluation phase of the M5 competition. Table 5 further depicts the forecasting error for the aggregated forecasts produced by LightGBM. The top-level N-BEATS models are evaluated with epochs of 10 and 12. The mean error is explicitly reported to check whether LightGBM over- or under-forecasts the actual values with the customized gradient in Eq. (1). Specifically, if the forecast is greater than the actual values, the error is positive, and vice versa. We ran the model with the number of epochs varying from 1 to 12. We noticed that it yielded better overall forecasting performance with 10 epochs, which was also used for the final submission of the M5 competition.

The code is written in Python, and reproducible Jupyter Notebooks running on the Kaggle platform are available at https://github.com/matthiasanderer/m5-accuracy-competition.

5. Ex post analysis

The M5 competition used a two-phase testing strategy consisting of the validation and evaluation phases. In the evaluation phase, no actual future time series values were available to the public during the competition. After the competition, the optimal value of the bias multiplier λ could be determined by looking at the overall WRMSSE metric in Eq. (4). We noticed that the intervals that contained the optimal multiplier were different in the two phases.

We checked the alignment performance in terms of forecasting errors with different bias multipliers for the validation timeframe with the actual disclosed future values. Fig. 2 shows the forecasting errors (WRMSSE) at all hierarchical levels based on the bottom level of LightGBM forecasts. One can observe that (i) the lower levels, which also contain larger portions of data, produce higher forecasting errors than the upper levels; and (ii) the overall WRMSSE on the validation timeframe reaches a minimum of 0.5291 when $\lambda = 1.16$. The loss multiplier λ was searched in the space of $(0, 2]$, and the WRMSSE was close to the result at the evaluation timeframe with WRMSSE $= 0.52816$. This indicates that by alternating the tuning parameter λ in the customized gradient function, one may lose some accuracy at the bottom level, but the upper levels' accuracy significantly improves. As a result, the overall accuracy is improved.

6. Conclusion

This paper proposed a hierarchical-forecasting-with-alignment approach that focuses on improving the point forecast accuracy for upper levels by aligning the high-accuracy top-level forecasts with the aggregated bottom-level forecasts in a hierarchical time series setting. The proposed top-down alignment approach ensures low forecasting errors on the upper levels of the hierarchy and improves the overall forecasting performance for an equally weighted metric like WRMSSE. Our research sheds light on an orthogonal direction for forecasting reconciliation, as suggested in, e.g., Wickramasuriya et al. (2019), allowing some suboptimal forecasts at the lower level while retaining the accuracy at upper levels.

The hierarchical-forecasting-with-alignment approach is straightforward to implement in practice. Improving overall forecasting performance requires accurate forecasting on the top level of the hierarchy. We employ the state-of-the-art deep learning forecasting approach, N-BEATS, for continuous time series at the top levels and a widely used tree-based algorithm, LightGBM, with non-time series features for the bottom-level intermittent time series. Both methods are easy to use and effortless to scale up with massive time series. It is worth mentioning that the proposed framework is general, and one could easily replace N-BEATS and LightGBM with other appropriate forecasting algorithms.

One notable difference compared to other approaches in the M5 competition is that the approach focuses on improving the forecasting accuracy on the continuous upper levels. We do not directly take the forecasting accuracy from bottom-level intermittent time series as our central attention, and the bottom-level forecasts are treated as mutable to ensure the hierarchical alignment. However, special combination techniques, e.g., Kang et al. (2021), could improve the accuracy of intermittent time series forecasting.

Although we focus on point forecasts in this paper, probabilistic forecasts with the proposed scheme should

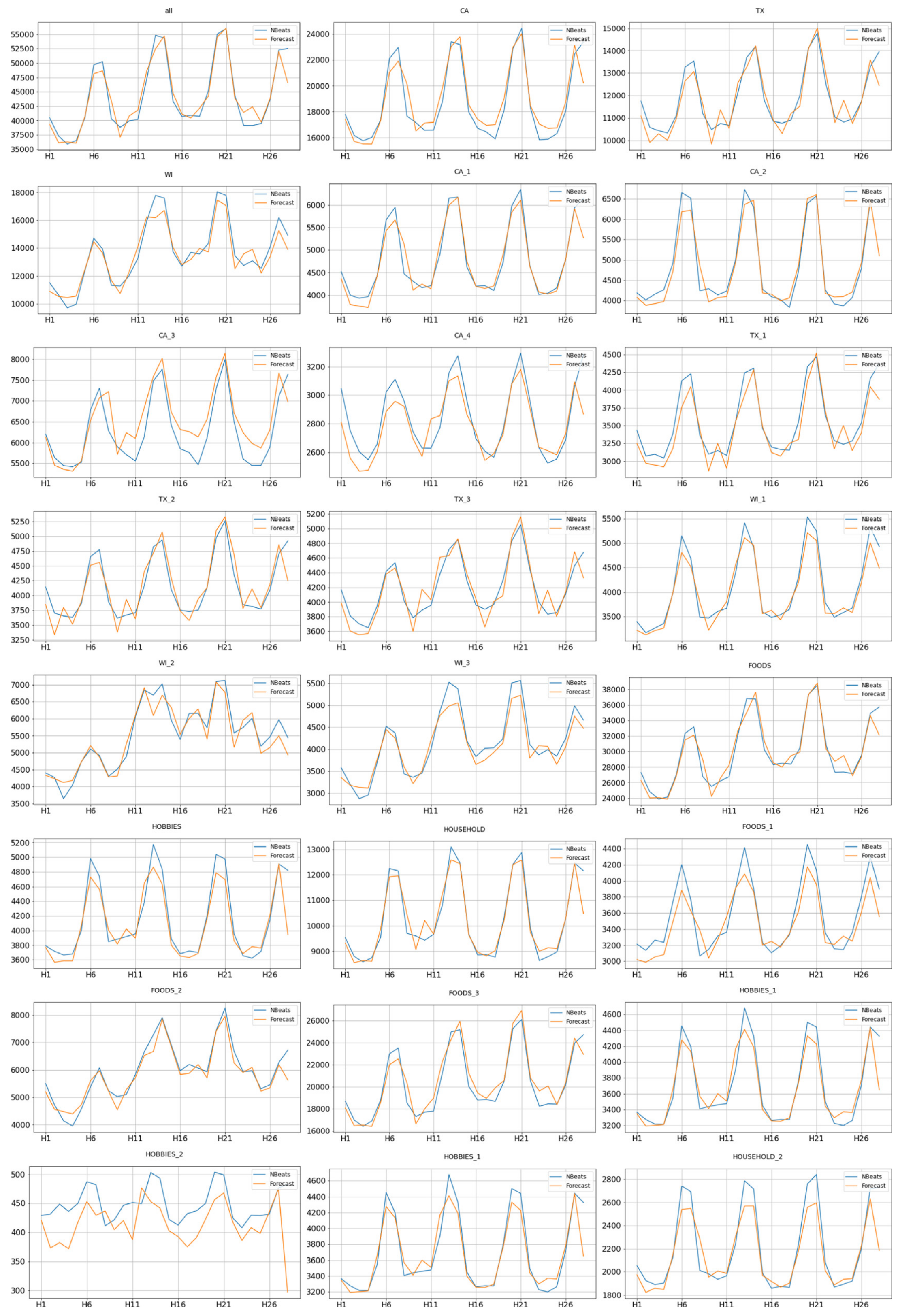

Fig. 1. Visualizing the N-BEATS forecasts (blue) and the bottom-up aggregated forecasts (orange) on the top five levels during the evaluation phase of the competition. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Hierarchical view of forecasting errors for the aggregated forecasts with LightGBM. The comparison is based on different accuracy metrics during the evaluation phase of the competition. The top-level N-BEATS models are evaluated with epochs of 10 and 12.

Hierarchy	Category	RMSSE		MEAN ERROR		MAE		RMSE	
		`epochs=10`	`epochs=12`	`epochs=10`	`epochs=12`	`epochs=10`	`epochs=12`	`epochs=10`	`epochs=12`
1	All	8.2742×10^{-5}	0.0031	−50.2829	305.5	1286.0	1379.9	1786.9	1891.1
2	CA	0.0038	0.0086	172.5	259.0	774.4	776.4	1011.3	1030.9
	TX	0.0073	7.6651×10^{-5}	−106.077	−10.9	405.2	387.4	510.3	510.8
	WI	0.0094	0.0114	−179.325	−197.3	580.9	544.0	669.4	665.3
3	CA_1	0.0065	0.0110	−62.3624	−80.9	165.8	176.7	252.0	276.0
	CA_2	0.0055	2.8249×10^{-6}	−76.8658	−1.7	225.1	217.8	338.4	313.0
	CA_3	0.1079	0.1569	247.1	297.9	373.8	425.1	437.4	493.6
	CA_4	0.0614	0.0344	−55.3935	−41.4	95.2	99.3	128.8	135.6
	TX_1	0.0606	0.0077	−108.084	−38.4	156.7	126.5	191.1	162.7
	TX_2	8.4627×10^{-6}	0.0018	−1.3454	19.8	176.7	168.5	222.5	225.9
	TX_3	0.0019	0.0015	−14.1593	12.6	133.3	137.4	161.8	164.5
	WI_1	0.0149	0.0022	−84.4941	−32.1	162.4	148.0	195.6	179.7
	WI_2	0.0097	0.0114	−64.7727	−70.3	225.5	242.8	282.4	301.9
	WI_3	0.0272	0.0203	−101.337	−87.6	186.9	143.2	219.1	179.8
4	FOODS	0.0002	0.0022	−44.8391	169.5	887.7	945.8	1190.4	1237.2
	HOBBIES	0.0265	0.0040	−80.9121	−31.4	145.7	128.2	220.2	200.8
	HOUSEHOLD	0.0006	0.0010	−35.4896	44.7	292.1	338.5	446.8	489.0
5	FOODS_1	0.0583	0.0463	−99.1094	−88.2	162.9	159.1	191.4	177.7
	FOODS_2	0.0256	0.0223	−117.115	−109.4	230.5	229.9	320.6	333.2
	FOODS_3	0.0084	0.0233	240.4	401.5	740.1	807.6	900.7	957.0
	HOBBIES_1	0.0046	0.0008	−31.4341	13.1	113.3	109.4	174.9	160.3
	HOBBIES_2	1.2066	1.2309	−35.1637	−35.5	39.2	38.4	51.7	52.1
	HOUSEHOLD_1	0.0003	0.0064	18.2	86.9	224.3	264.3	333.8	366.3
	HOUSEHOLD_2	0.0267	0.0175	−56.3958	−45.6	103.5	97.0	153.1	150.3

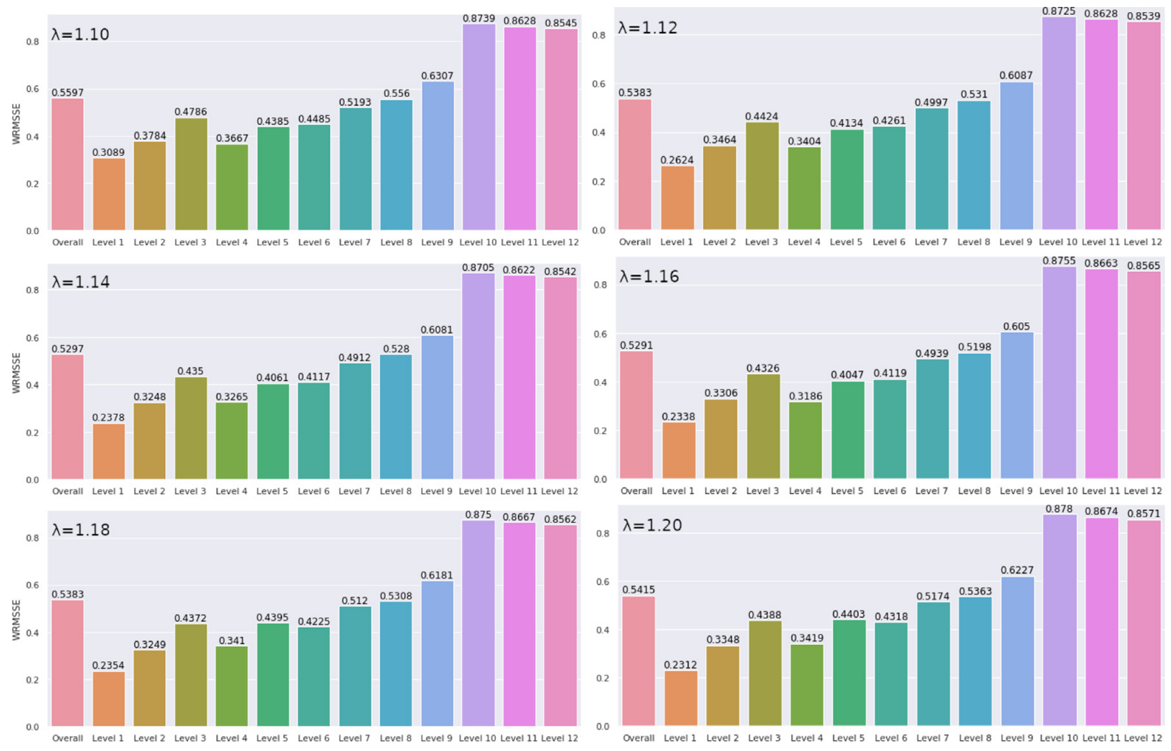

Fig. 2. Ex post analysis for the alignment between forecasting errors and the bias multiplier in the bottom-level model with the validation data of the competition. The number of epochs is 10 for the N-BEATS model on the top level.

be straightforward to implement with a corresponding probabilistic loss function. Another direction for future research is finding the best combination of top-level and bottom-level models. At the moment, forecasting for the upper levels and the bottom level is done independently. To utilize the information across hierarchical levels, we could in future studies consider a joint modeling scheme together with the alignment approach, or alignment with multiple levels. Combining the top-down alignment with other reconciliation methods is also possible but needs further investigation.

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

The authors are grateful to the editors and two anonymous reviewers for helpful comments that improved the contents of the paper. The authors also appreciate the public discussions on the time series features, models, and performance on the Kaggle open forum during the M5 competition. Feng Li is supported by the Beijing Universities Advanced Disciplines Initiative, China (No. GJJ2019163), the Emerging Interdisciplinary Project of CUFE, and the disciplinary funding of CUFE, China.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Tü, A. C., & Wang, Y. (2020). GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, *21*(116), 1–6, arXiv URL: http://jmlr.org/papers/v21/19-820.html.

Canera Turkmen, A., Januschowski, T., Wang, Y., & Taylana Cemgil, A. (2020). Intermittent demand forecasting with renewal processes. arXiv–2010, arXiv e-prints.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). ACM.

Croston, J. D. (1972). Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society*, *23*(3), 289–303, Taylor & Francis.

Gutierrez, R. S., Solis, A. O., & Mukhopadhyay, S. (2008). Lumpy demand forecasting using neural networks. *International Journal of Production Economics*, *111*(2), 409–420, Elsevier.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice, 2nd edition*. Melbourne, Australia: OTexts, OTexts.com/fpp2.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688, Elsevier.

Kang, Y., Cao, W., Petropoulos, F., & Li, F. (2021). Forecast with forecasts: Diversity matters. *European Journal of Operational Research*, http://dx.doi.org/10.1016/j.ejor.2021.10.024.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 3146–3154.

Kourentzes, N. (2013). Intermittent demand forecasts with neural networks. *International Journal of Production Economics*, *143*(1), 198–206, Elsevier.

Kourentzes, N., & Athanasopoulos, G. (2021). Elucidate structure in intermittent demand series. *European Journal of Operational Research*, *288*(1), 141–152, Elsevier.

Levé, E., & Segerstedt, A. (2004). Inventory control with a modified croston procedure and erlang distribution. *International Journal of Production Economics*, *90*(3), 361–367, Elsevier.

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. arXiv preprint arXiv:1912.09363.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021). The M5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*.

Nikolopoulos, K., Syntetos, A. A., Boylan, J. E., Petropoulos, F., & Assimakopoulos, V. (2011). An aggregate–disaggregate intermittent demand approach (ADIDA) to forecasting: an empirical proposition and analysis. *Journal of the Operational Research Society*, *62*(3), 544–554, Taylor & Francis.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, S. B., Bergmeir, C., Bessa, R. J., Bijak, J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Oliveira, F., Baets, S., Dokumentov, A., Ellison, J., ... Ziel, F. (2021). Forecasting: theory and practice. *International Journal of Forecasting*, In Press.

Rao, A. V. (1973). A comment on: forecasting and stock control for intermittent demands. *Journal of the Operational Research Society*, *24*(4), 639–640, Taylor & Francis.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191, Elsevier.

Shenstone, L., & Hyndman, R. J. (2005). Stochastic models underlying croston's method for intermittent demand forecasting. *Journal of Forecasting*, *24*(6), 389–402, Wiley Online Library.

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, Elsevier.

Syntetos, A. A., & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of Forecasting*, *21*(2), 303–314, Elsevier.

Wickramasuriya, S. L., Athanasopoulos, G., & Hyndman, R. J. (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, *114*(526), 804–819, Taylor & Francis.

Zhang, M., Lucas, J., Ba, J., & Hinton, G. E. (2019). Lookahead optimizer: k steps forward, 1 stepback. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, vol. 32*. Curran Associates, Inc.

Zufferey, T., Ulbig, A., Koch, S., & Hug, G. (2016). Forecasting of smart meter time series based on neural networks. In *International workshop on data analytics for renewable energy integration* (pp. 10–21). Springer.