N-BEATS: NEURAL BASIS EXPANSION ANALYSIS FOR INTERPRETABLE TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

We focus on solving the univariate times series point forecasting problem using deep learning. We propose a deep neural architecture based on backward and forward residual links and a very deep stack of fully-connected layers. The architecture has a number of desirable properties, being interpretable, applicable without modification to a wide array of target domains, and fast to train. We test the proposed architecture on several well-known datasets, including M3, M4 and TOURISM competition datasets containing time series from diverse domains. We demonstrate state-of-the-art performance for two configurations of N-BEATS for all the datasets, improving forecast accuracy by 11% over a statistical benchmark and by 3% over last year's winner of the M4 competition, a domain-adjusted hand-crafted hybrid between neural network and statistical time series models. The first configuration of our model does not employ any time-series-specific components and its performance on heterogeneous datasets strongly suggests that, contrarily to received wisdom, deep learning primitives such as residual blocks are by themselves sufficient to solve a wide range of forecasting problems. Finally, we demonstrate how the proposed architecture can be augmented to provide outputs that are interpretable without considerable loss in accuracy.

1 INTRODUCTION

Time series (TS) forecasting is an important business problem and a fruitful application area for machine learning (ML). It underlies most aspects of modern business, including such critical areas as inventory control and customer management, as well as business planning going from production and distribution to finance and marketing. As such, it has a considerable financial impact, often ranging in the millions of dollars for every point of forecasting accuracy gained (Jain, 2017; Kahn, 2003). And yet, unlike areas such as computer vision or natural language processing where deep learning (DL) techniques are now well entrenched, there still exists evidence that ML and DL struggle to outperform classical statistical TS forecasting approaches (Makridakis et al., 2018a;b). For instance, the rankings of the six "pure" ML methods submitted to M4 competition were 23, 37, 38, 48, 54, and 57 out of a total of 60 entries, and most of the best-ranking methods were ensembles of classical statistical techniques (Makridakis et al., 2018b).

On the other hand, the M4 competition winner, an approach by S. Smyl, was based on a hybrid between neural residual/attention dilated LSTM stack with a classical Holt-Winters statistical model (Holt, 1957; 2004; Winters, 1960) with learnable parameters. Since Smyl's approach heavily depends on this Holt-Winters component, Makridakis et al. (2018b) further argue that "hybrid approaches and combinations of method are the way forward for improving the forecasting accuracy and making forecasting more valuable". In this work we aspire to challenge this conclusion by exploring the potential of pure DL architectures in the context of the TS forecasting. Moreover, in the context of interpretable DL architecture design, we are interested in answering the following question: can we inject a suitable inductive bias in the model to make its internal operations more interpretable, in the sense of extracting some explainable driving factors combining to produce a given forecast?

1.1 SUMMARY OF CONTRIBUTIONS

Deep Neural Architecture: To the best of our knowledge, this is the first work to empirically demonstrate that pure DL using no time-series specific components outperforms well-established statistical approaches on M3, M4 and TOURISM datasets (on M4, by 11% over statistical benchmark, by 7% over the best statistical entry, and by 3% over the M4 competition winner). In our view, this provides a long-missing proof of concept for the use of pure ML in TS forecasting and strengthens motivation to continue advancing the research in this area.

Interpretable DL for Time Series: In addition to accuracy benefits, we also show that it is feasible to design an architecture with interpretable outputs that can be used by practitioners in very much the same way as traditional decomposition techniques such as the "seasonality-trend-level" approach (Cleveland et al., 1990).

2 Problem Statement

We consider the univariate point forecasting problem in discrete time. Given a length-H forecast horizon a length-T observed series history $[y_1, \dots, y_T] \in \mathbb{R}^T$, the task is to predict the vector of future values $\mathbf{y} \in \mathbb{R}^H = [y_{T+1}, y_{T+2}, \dots, y_{T+H}]$. For simplicity, we will later consider a *lookback window* of length $t \le T$ ending with the last observed value y_T to serve as model input, and denoted $\mathbf{x} \in \mathbb{R}^t = [y_{T-t+1}, \dots, y_T]$. We denote $\widehat{\mathbf{y}}$ the forecast of $\mathbf{y}$. The following metrics are commonly used to evaluate forecasting performance (Hyndman & Koehler, 2006; Makridakis & Hibon, 2000; Makridakis et al., 2018b; Athanasopoulos et al., 2011):

$$\text{sMAPE} = \frac{200}{H}\sum_{i=1}^{H}\frac{|y_{T+i} - \widehat{y}_{T+i}|}{|y_{T+i}| + |\widehat{y}_{T+i}|}, \qquad \text{MAPE} = \frac{100}{H}\sum_{i=1}^{H}\frac{|y_{T+i} - \widehat{y}_{T+i}|}{|y_{T+i}|},$$

$$\text{MASE} = \frac{1}{H}\sum_{i=1}^{H}\frac{|y_{T+i} - \widehat{y}_{T+i}|}{\frac{1}{T+H-m}\sum_{j=m+1}^{T+H}|y_j - y_{j-m}|}, \qquad \text{OWA} = \frac{1}{2}\left[\frac{\text{sMAPE}}{\text{sMAPE}_{\text{Naïve2}}} + \frac{\text{MASE}}{\text{MASE}_{\text{Naïve2}}}\right].$$

Here m is the periodicity of the data (*e.g.*, 12 for monthly series). MAPE (Mean Absolute Percentage Error), sMAPE (symmetric MAPE) and MASE (Mean Absolute Scaled Error) are standard scale-free metrics in the practice of forecasting (Hyndman & Koehler, 2006; Makridakis & Hibon, 2000): whereas sMAPE scales the error by the average between the forecast and ground truth, the MASE scales by the average error of the naïve predictor that simply copies the observation measured m periods in the past, thereby accounting for seasonality. OWA (overall weighted average) is a M4-specific metric used to rank competition entries (M4 Team, 2018b), where sMAPE and MASE metrics are normalized such that a seasonally-adjusted naïve forecast obtains OWA $= 1.0$.

3 N-BEATS

Our architecture design methodology relies on a few key principles. First, the base architecture should be simple and generic, yet expressive (deep). Second, the architecture should not rely on time-series-specific feature engineering or input scaling. These prerequisites let us explore the potential of pure DL architecture in TS forecasting. Finally, as a prerequisite to explore interpretability, the architecture should be extendable towards making its outputs human interpretable. We now discuss how those principles converge to the proposed architecture.

3.1 Basic Block

The proposed basic building block has a fork architecture and is depicted in Fig. 1 (left). It accepts an input $\mathbf{x}$ and outputs two vectors, $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{y}}$. Input $\mathbf{x}$ is a history lookback window of certain length ending with the last measured observation, $\widehat{\mathbf{y}}$ is the block's forward forecast of length H, and $\widehat{\mathbf{x}}$ is the block's best estimate of $\mathbf{x}$, also known as the 'backcast', given the constraints on the functional space that the block can use to approximate signals. We set the length of input window to a multiple of the forecast horizon H, and typical lengths of $\mathbf{x}$ in our setup range from $2H$ to $7H$.

Internally, the basic building block consists of two parts. First, the waveform generator $g_\theta : \mathscr{T}^N \mapsto \mathscr{Y}^N$ is a map from N points in the time domain $\mathscr{T}^N \subset \mathbb{R}^N$ to N points in the forecast (or backcast) value domain, $\mathscr{Y}^N \subset \mathbb{R}^N$. The number of points in forecast and backcast is different, in general. The waveform generator is parameterized with $\theta \in \Theta \subset \mathbb{R}^M$. The function of g_θ is to provide a sufficiently rich set of time-varying waveforms, selectable by varying θ. As shown below, g_θ can either be chosen to be learnable or can be set to specific funtional forms to reflect certain problem-specific

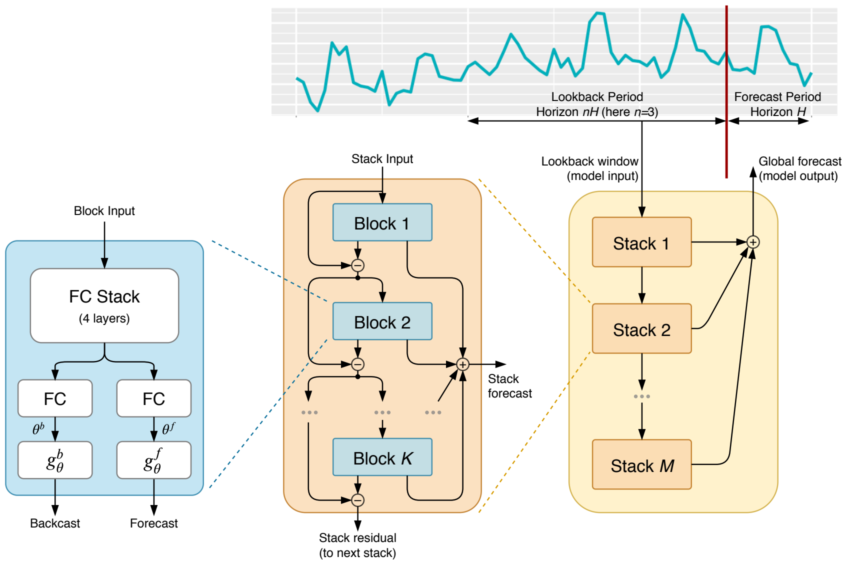


Figure 1: Proposed architecture. The basic building block is a multi-layer FC network with ReLU nonlinearities. It predicts basis expansion coefficients both forward, θ_f, (forecast) and backward, θ_b, (backcast). Blocks are organized into stacks using doubly residual stacking principle. A stack may have layers with shared g_θ. Forecasts are aggregated in hierarchical fashion. This enables building a very deep neural network with interpretable outputs.

inductive biases in order to appropriately constrain the structure of outputs. Concrete examples of g_θ are discussed in Section 3.3.

Second, the forward and the backward predictors of parameters of the waveform generator are maps $\varphi^f_\phi : \mathbb{R}^{\dim(\mathbf{x})} \mapsto \Theta$ and $\varphi^b_\phi : \mathbb{R}^{\dim(\mathbf{x})} \mapsto \Theta$. The task of φ^f_ϕ is to predict the forward expansion parameters θ_f with the ultimate goal of optimizing the accuracy of the partial forecast $\widehat{\mathbf{y}}$ by properly mixing the waveforms supplied by g_θ. Conversely, the task of φ^b_ϕ is to produce an estimate of $\mathbf{x}$ with the ultimate goal of helping the downstream blocks by removing components of their input that are not helpful for forecasting. We use φ_ϕ that has a form of the multi-layer fully connected (FC) neural network with ReLU non-linearity (Nair & Hinton, 2010). The number of hidden layers in the network is 4 and the width of each layer may be 256, 512 or 2048 units depending on where it is used (more details in Section 3.3 and Appendix E).

3.2 Doubly Residual Stacking

The classical well-known residual network architecture adds the input of the stack of layers to its output before passing the result to the next stack (He et al., 2016). The DenseNet architecture proposed by Huang et al. (2017) further extends this principle by introducing extra connections from the output of each layer stack to the input of every other layer stack that follows it. These approaches provide clear advantages in improving the trainability of very deep architectures. Their disadvantage in the context of this work is that they result in network structures that are difficult to interpret.

We propose a novel hierarchical doubly residual topology depicted in Fig. 1 (middle and right). The proposed architecture has two residual branches, one running over backcast prediction branch of each layer and the other one is running over the forecast branch of each layer. The backcast residual branch can be thought of as running a sequential analysis of the input signal. Each block removes the portion of the signal that it can explain well, making the forecast job of the downstream blocks easier. This structure also facilitates more fluid gradient backpropagation. More importantly, each layer outputs a partial forecast that is first aggregated at the stack level and then at the overall network level, providing a hierarchical decomposition. In a generic model context, when stacks are allowed to have arbitrary g_θ for each layer, this does not have an effect other than making the network more transparent to gradient flows. In a special situation of deliberate structure enforced in g_θ shared over a stack, explained next, this has the critical importance of enabling interpretability via the aggregation of meaningful partial forecasts.

3.3 INTERPRETABILITY

We propose two configurations of the architecture, based on the selection of g_θ. One of them is generic DL, the other one is augmented with certain inductive biases to be interpretable.

The **generic architecture** does not rely on TS-specific knowledge. We set g_θ to be a linear projection of the previous layer output. In this case the partial forecast at the output of block j in stack i is:

$$\widehat{\mathbf{y}}_{i,j} = \mathbf{W}_{i,j}\theta_{i,j} + \mathbf{b}_{i,j}.$$

The interpretation of this model is that the FC layers in the basic building block depicted in Fig. 1 learn the predictive decomposition of the partial forecast $\widehat{\mathbf{y}}_{i,j}$ in the basis $\mathbf{W}_{i,j}$ learned by the network. Matrix $\mathbf{W}_{i,j}$ has dimensionality $H \times \dim(\theta_{i,j})$. Therefore, the first dimension of $\mathbf{W}_{i,j}$ has the interpretation of discrete time index in the forecast domain. The second dimension of the matrix has the interpretation of the indices of the basis functions, with $\theta_{i,j}$ being the expansion coefficients for this basis. Thus the columns of $\mathbf{W}_{i,j}$ can be thought of as waveforms in the time domain. Because no additional constraints are imposed on the form of $\mathbf{W}_{i,j}$, the waveforms learned by the deep model do not have inherent structure (and none is apparent in our experiments). This leads to $\widehat{\mathbf{y}}_{i,j}$ as well as stack outputs $\widehat{\mathbf{y}}_i = \sum_j \widehat{\mathbf{y}}_{i,j}$ not being interpretable.

The **interpretable architecture** can be constructed by reusing the overall architectural approach in Fig. 1 and by adding structure to g_θ for each stack. Forecasting practitioners often use the decomposition of time series into trend and seasonality, such as those performed by the STL (Cleveland et al., 1990) and X13-ARIMA (U.S. Census Bureau, 2013). We propose to design the trend and seasonality decomposition into the model to make the stack outputs more easily interpretable.

Trend model. A typical characteristic of trend is that most of the time it is a monotonic function, or at least a slowly varying function. In order to mimic this behaviour we propose to constrain g_θ to be a polynomial of small degree p, a function slowly varying across forecast window:

$$g_\theta(t) = \sum_{i=0}^{p} \theta_i t^i. \tag{1}$$

Here time vector $\mathbf{t} = [0,1,2,\dots,H-2,H-1]^T/H$ is defined on a discrete grid running from 0 to $(H-1)/H$, forecasting H steps ahead. The trend forecast will then have the form:

$$\widehat{\mathbf{y}}^{tr}_{i,j} = \mathbf{T}\theta_{i,j},$$

where $\theta_{i,j}$ are polynomial coefficients predicted by a FC network of layer j of stack i and $\mathbf{T} = [\mathbf{1}, \mathbf{t}, \dots, \mathbf{t}^p]$ is the matrix of powers of $\mathbf{t}$. If p is low, e.g. 2 or 3, it forces $\widehat{\mathbf{y}}^{tr}_{i,j}$ to mimic trend.

Seasonality model. Typical characteristic of seasonality is that it is a regular, cyclical, recurring fluctuation. Therefore, to model seasonality, we propose to constrain g_θ to belong to the class of periodic functions $g_\theta(t) = g_\theta(t-s)$, where s is a seasonality period. A natural choice for the basis to model periodic function is the Fourier series:

$$g_\theta(t) = \sum_{i=0}^{\lfloor H/2-1 \rfloor} \theta_i \cos(2\pi i t) + \theta_{i+\lfloor H/2 \rfloor} \sin(2\pi i t), \tag{2}$$

The seasonality forecast will then have the form:

$$\widehat{\mathbf{y}}^{s}_{i,j} = \mathbf{S}\theta_{i,j},$$

where $\theta_{i,j}$ are Fourier coefficients predicted by a FC network of layer j of stack i and $\mathbf{S} = [\mathbf{1}, \cos(2\pi\mathbf{t}), \dots \cos(2\pi\lfloor H/2-1 \rfloor \mathbf{t})), \sin(2\pi\mathbf{t}), \dots, \sin(2\pi\lfloor H/2-1 \rfloor \mathbf{t}))]$ is the matrix of sinusoidal waveforms. The forecast $\widehat{\mathbf{y}}^{s}_{i,j}$ is then a periodic function mimicking typical seasonal patterns.

The overall interpretable architecture consists of two stacks only: the trend modeling stack is followed by the seasonality modeling stack. The doubly residual stacking combined with the forecast/backcast principle result in (i) the trend component being removed from the input window $\mathbf{x}$ before it is fed into the seasonality analysis stack and (ii) the partial forecasts of trend and seasonality are available as separate interpretable outputs. Structurally, each of the stacks consists of several blocks connected with residual connections as depicted in Fig. 1 and each of them shares its respective, non-learnable g_θ. The number of blocks is 3 for both trend and seasonality. We found that on top of sharing g_θ, sharing all the weights across blocks in a stack resulted in better performance on validation set.

3.4 Ensembling

Ensembling is used by all the top entries in the M4-competition. We rely on ensembling as well to be comparable. We found that ensembling is a much more powerful regularization technique than the popular alternatives, e.g. dropout or L2-norm penalty. The addition of those methods improved individual models, but was hurting the performance of the ensemble. The core property of an ensemble is diversity. We build an ensemble using several sources of diversity. First, the ensemble models are fit on three different metrics: sMAPE, MASE and MAPE, a version of sMAPE that has only the ground truth value in the denominator. Second, for every horizon H, individual models are trained on input windows of different length: $2H, 3H, \dots, 7H$, for a total of six window lengths. Thus the overall ensemble exhibits a multi-scale aspect. Finally, we perform a bagging procedure (Breiman, 1996) by including models trained with different random initializations. We use 180 total models to report results on the test set (please refer to Appendix C for the ablation of ensemble size). We use the median as ensemble aggregation function.

4 Related Work

The approaches to TS forecasting can be split in a few distinct categories. The statistical modeling approaches based on exponential smoothing and its different flavors are well established and are often considered a default choice in the industry (Holt, 1957; 2004; Winters, 1960). More advanced variations of exponential smoothing include the winner of M3 competition, the Theta method (Assimakopoulos & Nikolopoulos, 2000) that decomposes the forecast into several theta-lines and statistically combines them. The pinnacle of the statistical approach encapsulates ARIMA, auto-ARIMA and in general, the unified state-space modeling approach, that can be used to explain and analyze all of the approaches mentioned above (see Hyndman & Khandakar (2008) for an overview). More recently, ML/TS combination approaches started infiltrating the domain with great success, showing promising results by using the outputs of statistical engines as features. In fact, 2 out of top-5 entries in the M4 competition are approaches of this type, including the second entry (Montero-Manso et al., 2019). The second entry computes the outputs of several statistical methods on the M4 dataset and combines them using gradient boosted tree (Chen & Guestrin, 2016). Somewhat independently, the work in the modern deep learning TS forecasting developed based on variations of recurrent neural networks (Flunkert et al., 2017; Rangapuram et al., 2018; Toubeau et al., 2019; Zia & Razzaq, 2018) being largely dominated by the electricity load forecasting in the multi-variate setup. A few earlier works explored the combinations of recurrent neural networks with dilation, residual connections and attention (Chang et al., 2017; Kim et al., 2017; Qin et al., 2017). These served as a basis for the winner of the M4 competition, an approach developed by S. Smyl. The winning entry combines a Holt-Winters style seasonality model with its parameters fitted to a given TS via gradient descent and a unique combination of dilation/residual/attention approaches for each forecast horizon. The resulting model is a hybrid model that architecturally heavily relies on a time-series engine. It is hand crafted to each specific horizon of M4, making this approach hard to generalize to other datasets.

5 Experimental Results

Our key empirical results based on aggregate performance metrics over several datasets—M4 (M4 Team, 2018b; Makridakis et al., 2018b), M3 (Makridakis & Hibon, 2000; Makridakis et al., 2018a) and TOURISM (Athanasopoulos et al., 2011)—appear in Table 1. More detailed descriptions of the datasets are provided in Section 5.1 and Appendix B. For each dataset, we compare our results with best 5 entries for this dataset reported in the literature, according to the customary metrics specific to each dataset (M4: OWA and sMAPE, M3: sMAPE, TOURISM: MAPE). More granular dataset-specific results with data splits over forecast horizons and types of time series appear in respective appendices (M4: Appendix D.1; M3: Appendix D.2; TOURISM: Appendix D.3).

In Table 1, we study the performance of two N-BEATS configurations: generic (N-BEATS-G) and interpretable (N-BEATS-I), as well as N-BEATS-I+G (ensemble of all models from N-BEATS-G and N-BEATS-I). **On M4 dataset**, we compare against 5 representatives from the M4 competition (Makridakis et al., 2018b): each best in their respective model class. *Pure ML* is the submission by B. Trotta, the best entry among the 6 pure ML models. *Statistical* is the best pure statistical model by N.Z. Legaki and K. Koutsouri. *ML/TS combination* is the model by P. Montero-Manso,

Table 1: Performance on the M4, M3, TOURISM test sets, aggregated over each dataset. Evaluation metrics are specified for each dataset; lower values are better. The number of time series in each dataset is provided in brackets.

M4 Average (100,000)			M3 Average (3,003)		TOURISM Average (1,311)	
	SMAPE	OWA		SMAPE		MAPE
Pure ML	12.894	0.915	Comb S-H-D	13.52	ETS	20.88
Statistical	11.986	0.861	ForecastPro	13.19	Theta	20.88
ProLogistica	11.845	0.841	Theta	13.01	ForePro	19.84
ML/TS combination	11.720	0.838	DOTM	12.90	Stratometrics	19.52
DL/TS hybrid	11.374	0.821	EXP	12.71	LeeCBaker	19.35
N-BEATS-G	11.168	0.797		12.47		**18.47**
N-BEATS-I	11.174	0.798		12.43		18.97
N-BEATS-I+G	**11.135**	**0.795**		**12.37**		18.52

T. Talagala, R.J. Hyndman and G. Athanasopoulos, second best entry, gradient boosted tree over a few statistical time series models. ProLogistica is the third entry in M4 based on the weighted ensemble of statistical methods. Finally, *DL/TS hybrid* is the winner of M4 competition designed by S. Smyl. **On the M3 dataset**, we compare against the *Theta* method (Assimakopoulos & Nikolopoulos, 2000), the winner of M3; *DOTA*, a dynamically optimized Theta model (Fiorucci et al., 2016); *EXP*, the most resent statistical approach and the previous state-of-the-art on M3 (Spiliotis et al., 2019); as well as *ForecastPro*, an off-the-shelf forecasting software that is based on model selection between exponential smoothing, ARIMA and moving average (Athanasopoulos et al., 2011; Assimakopoulos & Nikolopoulos, 2000). **On the TOURISM dataset**, we compare against 3 statistical benchmarks (Athanasopoulos et al., 2011): *ETS*, exponential smoothing with cross-validated additive/multiplicative model; *Theta* method; *ForePro*, same as *ForecastPro* in M3; as well as top 2 entries from the TOURISM Kaggle competition (Athanasopoulos & Hyndman, 2011): *Stratometrics*, an unknown technique; *LeeCBaker* (Baker & Howard, 2011), a weighted combination of Naïve, linear trend model, and exponentially weighted least squares regression trend.

According to Table 1, N-BEATS demonstrates state-of-the-art performance on three challenging non-overlapping datasets containing time series from very different domains, sampling frequencies and seasonalities. As an example, on M4 dataset, the OWA gap between N-BEATS and the M4 winner ($0.821 - 0.795 = 0.026$) is greater than the gap between the M4 winner and the second entry ($0.838 - 0.821 = 0.017$). Generic N-BEATS model uses as little prior knowledge as possible, with no feature engineering, no scaling and no internal architectural components that may be considered TS-specific. Thus the result in Table 1 leads us to the conclusion that DL does not need support from the statistical approaches or hand-crafted feature engineering and domain knowledge to perform extremely well on a wide array of TS forecasting tasks. On top of that, the proposed general architecture performs very well on three different datasets outperforming a wide variety of models, both generic and manually crafted to respective dataset, including the winner of M4, a model architecturally adjusted by hand to each forecast-horizon subset of the M4 data.

5.1 Datasets

M4 (M4 Team, 2018b; Makridakis et al., 2018b) is the latest in an influential series of forecasting competitions organized by Spyros Makridakis since 1982 (Makridakis et al., 1982). The 100k-series dataset is large and diverse, consisting of data frequently encountered in business, financial and economic forecasting, and sampling frequencies ranging from hourly to yearly. A table with summary statistics is presented in Appendix B.1, showing wide variability in TS characteristics.

M3 (Makridakis & Hibon, 2000) is similar in its composition to M4, but has a smaller overall scale (3003 time series total vs. 100k in M4). A table with summary statistics is presented in Appendix B.2. Over the past 20 years, this dataset has supported significant efforts in the design of more optimal statistical models, e.g. Theta and its variants (Assimakopoulos & Nikolopoulos, 2000; Fiorucci et al., 2016; Spiliotis et al., 2019). Furthermore, a recent publication (Makridakis et al., 2018a) based on a subset of M3 presented evidence that ML models are inferior to the classical statistical models.

TOURISM (Athanasopoulos et al., 2011) dataset was released as part of the respective Kaggle competition conducted by Athanasopoulos & Hyndman (2011). The data include monthly, quarterly and yearly series supplied by both governmental tourism organizations (e.g. Tourism Australia, the Hong Kong Tourism Board and Tourism New Zealand) as well as various academics, who had used them in previous studies. A table with summary statistics is presented in Appendix B.3.

5.2 Training methodology

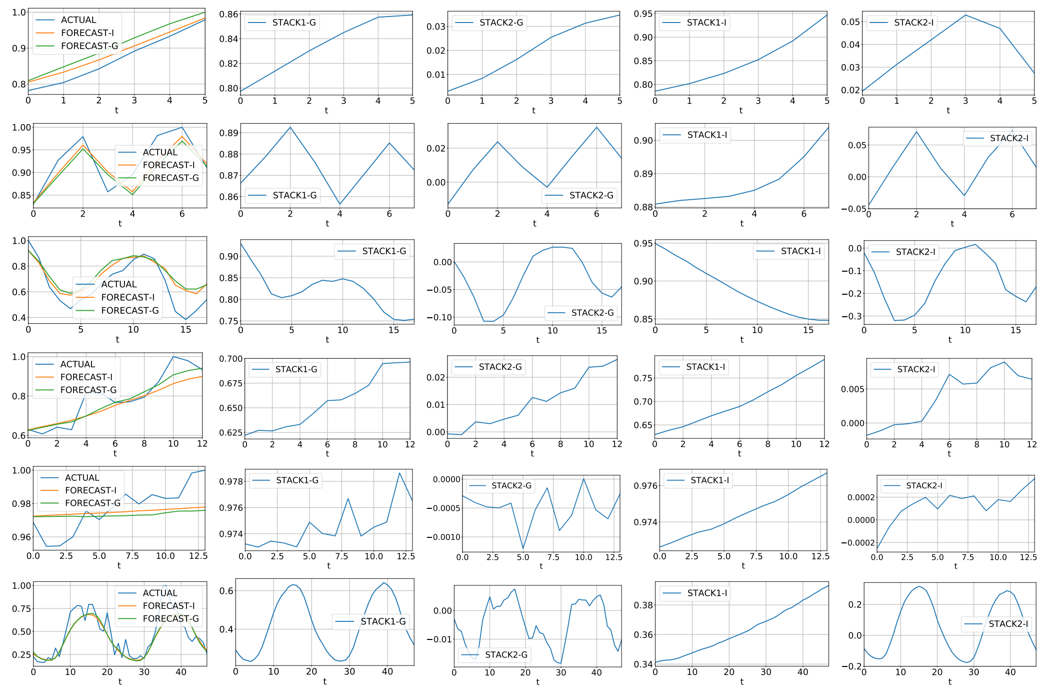
We split each dataset into train, validation and test subsets. The test subset is the standard test set previously defined for each dataset (M4 Team, 2018a; Makridakis & Hibon, 2000; Athanasopoulos et al., 2011). The validation and train subsets for each dataset are obtained by splitting their full train sets at the boundary of the last horizon of each time series. We use the train and validation subsets to tune hyperparameters. For M4, we also use validation set to report the results of ablation studies (see Appendix C). Once the hyperparameters are determined, we train the model on the full train set and report results on the test set. N-BEATS is implemented and trained in Tensorflow (Abadi et al., 2015).

We share parameters of the network across horizons, therefore we train one model per horizon for each dataset. If every time series is interpreted as a separate task, this can be linked back to the multitask learning and furthermore to meta-learning (see discussion in Appendix A), in which a neural network is regularized by learning on multiple tasks to improve generalization. We would like to stress that models for different horizons and datasets reuse the same architecture. Architectural hyperparameters (width, number of layers, number of stacks, etc.) are fixed to the same values across horizons and across datasets (see Appendix E). The fact that we can reuse architecture and even hyperparameters across horizons indicates that the proposed architecture design generalizes well across time series of different nature. The same architecture is successfully trained on the M4 Monthly subset with 48k time series and the M3 Others subset with 174 time series. This is a much stronger result than *e.g.* the result of S. Smyl (Makridakis et al., 2018b) who had to use very different architectures hand crafted for different horizons.

To update network parameters for one horizon, we sample train batches of fixed size 1024. We pick 1024 TS ids from this horizon, uniformly at random with replacement. For each selected TS id we pick a random forecast point from the historical range of length L_H immediately preceding the last point in the train part of the TS. L_H is a cross-validated hyperparameter. We observed that for subsets with large number of time series it tends to be smaller and for subsets with smaller number of time series it tends to be larger. For example, in massive Yearly, Monthly, Quarterly subsets of M4 L_H is equal to 1.5; and in moderate to small Weekly, Daily, Hourly subsets of M4 L_H is equal to 10. Given a sampled forecast point, we set one horizon worth of points following it to be the target forecast window $\mathbf{y}$ and we set the history of points of one of lengths $2H, 3H, \dots, 7H$ preceding it to be the input $\mathbf{x}$ to the network. We use the Adam optimizer with default settings and initial learning rate 0.001. The neural network training is run with early stopping and the number of batches is determined on the validation set. The GPU based training of one ensemble member for entire M4 dataset takes between 30 min and 2 hours depending on neural network settings and hardware.

5.3 Interpretability results

Fig. 2 studies the outputs of the proposed model in the generic and the interpretable configurations. As discussed in Section 3.3, to make the generic architecture presented in Fig. 1 interpretable, we constrain g_θ in the first stack to have the form of polynomial (1) while the second one has the form of Fourier basis (2). Furthermore, we use the outputs of the generic configuration of N-BEATS as control group (the generic model of 30 residual blocks depicted in Fig. 1 is divided into two stacks) and we plot both generic (suffix “-G”) and interpretable (suffix “-I”) stack outputs side by side in Fig. 2. The outputs of generic model are arbitrary and non-interpretable: either trend or seasonality or both of them are present at the output of both stacks. The magnitude of the output (peak-to-peak) is generally smaller at the output of the second stack. The outputs of the interpretable model exhibit distinct properties: the trend output is monotonic and slowly moving, the seasonality output is regular, cyclical and has recurring fluctuations. The peak-to-peak magnitude of the seasonality output is significantly larger than that of the trend, if significant seasonality is present in the time series. Similarly, the peak-to-peak magnitude of trend output tends to be small when no obvious trend is present in the ground truth signal. Thus the proposed interpretable architecture decomposes its



(a) Combined (b) Stack1-G (c) Stack2-G (d) StackT-I (e) StackS-I

Figure 2: The outputs of generic and the interpretable configurations, M4 dataset. Each row is one time series example per data frequency, top to bottom (Yearly: id Y3974, Quarterly: id Q11588, Monthly: id M19006, Weekly: id W246, Daily: id D404, Hourly: id H344). The magnitudes in a row are normalized by the maximal value of the actual time series for convenience. Column (a) shows the actual values (ACTUAL), the generic model forecast (FORECAST-G) and the interpretable model forecast (FORECAST-I). Columns (b) and (c) show the outputs of stacks 1 and 2 of the generic model, respectively; FORECAST-G is their summation. Columns (d) and (e) show the output of the Trend and the Seasonality stacks of the interpretable model, respectively; FORECAST-I is their summation.

forecast into two distinct components. Our conclusion is that the outputs of the DL model can be made interpretable by encoding a sensible inductive bias in the architecture. Table 1 confirms that this does not result in performance drop.

6 Conclusions

We proposed and empirically validated a novel architecture for univariate TS forecasting. We showed that the architecture is general, flexible and it performs well on a wide array of TS forecasting problems. We applied it to three non-overlapping challenging competition datasets: M4, M3 and TOURISM and demonstrated state-of-the-art performance in two configurations: generic and interpretable. This allowed us to validate two important hypotheses: (i) the generic DL approach performs exceptionally well on heterogeneous univariate TS forecasting problems using no TS domain knowledge, (ii) it is viable to additionally constrain a DL model to force it to decompose its forecast into distinct human interpretable outputs. We also demonstrated that the DL models can be trained on multiple time series in a multi-task fashion, successfully transferring and sharing individual learnings. We speculate that N-BEATS's performance can be attributed in part to it carrying out a form of meta-learning, a deeper investigation of which should be the subject of future work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew

Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

V. Assimakopoulos and K. Nikolopoulos. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, 2000.

George Athanasopoulos and Rob J. Hyndman. The value of feedback in forecasting competitions. *International Journal of Forecasting*, 27(3):845–849, 2011.

George Athanasopoulos, Rob J. Hyndman, Haiyan Song, and Doris C. Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011.

Lee C. Baker and Jeremy Howard. Winning methods for forecasting tourism time series. *International Journal of Forecasting*, 27(3):850–852, 2011.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. In *Proceedings of the International Joint Conference on Neural Networks*, pp. II–A969, Seattle, USA, 1991.

Christoph Bergmeir, Rob J. Hyndman, and José M. Benítez. Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. *International Journal of Forecasting*, 32 (2):303–312, 2016.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

Phil Brierley. Winning methods for forecasting seasonal tourism time series. *International Journal of Forecasting*, 27(3):853–854, 2011.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In *NIPS*, pp. 77–87, 2017.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD*, pp. 785–794, 2016.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A seasonal-trend decomposition procedure based on Loess (with discussion). *Journal of Official Statistics*, 6: 3–73, 1990.

Jose A. Fiorucci, Tiago R. Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B. Koehler. Models for optimising the Theta method and their relationship to state space models. *International Journal of Forecasting*, 32(4):1151–1161, 2016.

Valentin Flunkert, David Salinas, and Jan Gasthaus. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

C. C. Holt. Forecasting trends and seasonals by exponentially weighted averages. Technical Report ONR memorandum no. 5, Carnegie Institute of Technology, Pittsburgh, PA, 1957.

Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pp. 2261–2269. IEEE Computer Society, 2017.

Rob Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

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

Chaman L. Jain. Answers to your forecasting questions. *Journal of Business Forecasting*, 36, Spring 2017.

Kenneth B. Kahn. How to measure the impact of a forecast error on an enterprise? *The Journal of Business Forecasting Methods & Systems*, 22(1), Spring 2003.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. In *Interspeech 2017*, pp. 1591–1595, 2017.

M4 Team. M4 dataset, 2018a. URL `https://github.com/M4Competition/M4-methods/tree/master/Dataset`.

M4 Team. M4 competitor's guide: prizes and rules, 2018b. URL `www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-CompetitorsGuide.pdf`.

S Makridakis, E Spiliotis, and V Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13(3), 2018a.

Spyros Makridakis and Michèle Hibon. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000.

Spyros Makridakis, A Andersen, Robert Carbone, Robert Fildes, Michele Hibon, Rudolf Lewandowski, Joseph Newton, Emanuel Parzen, and Robert Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2): 111–153, 1982.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4-Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4): 802–808, 2018b.

Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. FFORMA: Feature-based Forecast Model Averaging. *International Journal of Forecasting*, 2019. to appear.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pp. 807–814, 2010.

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI-17*, pp. 2627–2633, 2017.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *NeurIPS 31*, pp. 7785–7794, 2018.

Slawek Smyl and Karthik Kuber. Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In *36th International Symposium on Forecasting*, 2016.

Evangelos Spiliotis, Vassilios Assimakopoulos, and Konstantinos Nikolopoulos. Forecasting with a hybrid method utilizing data smoothing, a variation of the theta method and shrinkage of seasonal factors. *International Journal of Production Economics*, 209:92–102, 2019.

A. A. Syntetos, J. E. Boylan, and J. D. Croston. On the categorization of demand patterns. *Journal of the Operational Research Society*, 56(5):495–503, 2005.

J. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34 (2):1203–1215, March 2019.

U.S. Census Bureau. Reference manual for the X-13ARIMA-SEATS Program, version 1.0, 2013. URL `http://www.census.gov/ts/x13as/docX13AS.pdf`.

Peter R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960.

Tehseen Zia and Saad Razzaq. Residual recurrent highway networks for learning deep sequence prediction models. *Journal of Grid Computing*, Jun 2018.

A Discussion: Connections to Meta-learning

Meta-learning defines an inner *learning procedure* and an outer *learning procedure*. The inner learning procedure is parameterized, conditioned or otherwise influenced by the outer learning procedure (Bengio et al., 1991). The prototypical inner vs. outer learning is individual learning in the lifetime of an animal vs. evolution of the inner learning procedure itself over many generations of individuals. To see the two levels, it often helps to refer to two sets of parameters, the inner parameters (e.g. synaptic weights) which are modified inside the inner learning procedure, and the outer parameters or meta-parameters (e.g. genes) which are get modified only in the outer learning procedure.

N-BEATS can be cast as an instance of meta-learning by drawing the following parallels. The outer learning procedure is encapsulated in the parameters of the whole network, learned by gradient descent. The inner learning procedure is encapsulated in the set of basic building blocks and modifies the parameters θ of g_θ. The inner learning proceeds through a sequence of stages, each corresponding to a block within the stack of the architecture. Each of the blocks can be thought of as performing the equivalent of an update step which gradually modifies the parameters θ which eventually feed into g_θ in each block (which get added together to form the final prediction). The inner learning procedure takes a single history from a piece of a TS and sees that history as a training set. It produces forward parameters θ^f (see Fig. 1), which parametrically map inputs to predictions. In addition, each preceding block modifies the input to the next block by producing backward parameters θ^b, thus conditioning the learning and the output of the next block. In the case of the interpretable model, the meta-parameters are only in the FC layers because the g_θ's are fixed. In the case of the generic model, the meta-parameters also include the $\mathbf{W}$'s which define the g_θ non-parametrically. This point of view is further reinforced by the results of the ablation study reported in Appendix C showing that increasing the number of blocks in the stack, as well as the number of stacks improves generalization performance, and can be interpreted as more iterations of the inner learning procedure.

B Dataset Details

B.1 M4 Dataset Details

Table 2 outlines the composition of the M4 dataset across domains and forecast horizons by listing the number of time series based on their frequency and type (M4 Team, 2018b). The M4 dataset is large and diverse: all forecast horizons are composed of heterogeneous time series types (with exception of Hourly) frequently encountered in business, financial and economic forecasting. Summary statistics on series lengths are also listed, showing wide variability therein, as well as a characterization (*smooth* vs *erratic*) that follows Syntetos et al. (2005), and is based on the squared coefficient of variation of the series. All series have positive observed values at all time-steps; as such, none can be considered *intermittent* or *lumpy* per Syntetos et al. (2005).

B.2 M3 Dataset Details

Table 3 outlines the composition of the M3 dataset across domains and forecast horizons by listing the number of time series based on their frequency and type (Makridakis & Hibon, 2000). The M3 is smaller than the M4, but it is still large and diverse: all forecast horizons are composed of heterogeneous time series types frequently encountered in business, financial and economic forecasting. Summary statistics on series lengths are also listed, showing wide variability in length, as well as a characterization (*smooth* vs *erratic*) that follows Syntetos et al. (2005), and is based on the squared coefficient of variation of the series. All series have positive observed values at all time-steps; as such, none can be considered *intermittent* or *lumpy* per Syntetos et al. (2005).

B.3 tourism Dataset Details

Table 4 outlines the composition of the tourism dataset across forecast horizons by listing the number of time series based on their frequency. Summary statistics on series lengths are listed, showing wide variability in length. All series have positive observed values at all time-steps. In contrast to M4 and M3 datasets, tourism includes a much higher fraction of erratic series.

Table 2: Composition of the M4 dataset: the number of time series based on their sampling frequency and type.

	Frequency / Horizon						
Type	Yearly/6	Qtly/8	Monthly/18	Wkly/13	Daily/14	Hrly/48	Total
Demographic	1,088	1,858	5,728	24	10	0	8,708
Finance	6,519	5,305	10,987	164	1,559	0	24,534
Industry	3,716	4,637	10,017	6	422	0	18,798
Macro	3,903	5,315	10,016	41	127	0	19,402
Micro	6,538	6,020	10,975	112	1,476	0	25,121
Other	1,236	865	277	12	633	414	3,437
Total	23,000	24,000	48,000	359	4,227	414	100,000
Min. Length	19	24	60	93	107	748	
Max. Length	841	874	2812	2610	9933	1008	
Mean Length	37.3	100.2	234.3	1035.0	2371.4	901.9	
SD Length	24.5	51.1	137.4	707.1	1756.6	127.9	
% Smooth	82%	89%	94%	84%	98%	83%	
% Erratic	18%	11%	6%	16%	2%	17%	

Table 3: Composition of the M3 dataset: the number of time series based on their sampling frequency and type.

	Frequency / Horizon				
Type	Yearly/6	Quarterly/8	Monthly/18	Other/8	Total
Demographic	245	57	111	0	413
Finance	58	76	145	29	308
Industry	102	83	334	0	519
Macro	83	336	312	0	731
Micro	146	204	474	4	828
Other	11	0	52	141	204
Total	645	756	1,428	174	3,003
Min. Length	20	24	66	71	
Max. Length	47	72	144	104	
Mean Length	28.4	48.9	117.3	76.6	
SD Length	9.9	10.6	28.5	10.9	
% Smooth	90%	99%	98%	100%	
% Erratic	10%	1%	2%	0%	

Table 4: Composition of the TOURISM dataset: the number of time series based on their sampling frequency.

	Frequency / Horizon			
	Yearly/4	Quarterly/8	Monthly/24	Total
	518	427	366	1,311
Min. Length	11	30	91	
Max. Length	47	130	333	
Mean Length	24.4	99.6	298	
SD Length	5.5	20.3	55.7	
% Smooth	77%	61%	49%	
% Erratic	23%	39%	51%	

Table 5: sMAPE on the validation set, generic architecture. sMAPE for varying number of stacks, each having one residual block.

Stacks	sMAPE
1	11.154
3	11.061
9	10.998
18	10.950
30	10.937

Table 6: sMAPE on the validation set, interpretable architecture. Ablation of the synergy of the layers with different basis functions and multi-block stack gain.

Detrend	Seasonality	sMAPE
0	2	11.189
2	0	11.572
1	1	11.040
3	3	10.986

C Ablation Studies

We performed an ablation study on the validation set, using sMAPE metric as performance criterion. We addressed two specific questions with this study. First, Is stacking layers helpful? Second, Does the architecture based on the combination of layers with different basis functions results in better performance than the architecture using only one layer type?

Layer stacking. We start our study with the generic architecture that consists of stacks of one residual block of 5 FC layers each of the form Fig. 1 and we increase the number of stacks. Results presented in Table 5 confirm that increasing the number of stacks decreases error and at certain point the gain saturates. We would like to mention that the network having 30 stack of depth 5 is in fact a very deep network of total depth 150 layers.

Basis synergy. Stacking works well for the interpretable architecture as can be seen in Table 6 depicting the results of ablating the interpretable architecture configuration. Here we experiment with the architecture that is composed of 2 stacks, stack one is trend model and stack two is the seasonality model. Each stack has variable number of residual blocks and each residual block has 5 FC layers. We found that this architecture works best when all weights are shared within stack. We clearly see that increasing the number of layers improves performance. The largest network is 60 layers deep. On top of that, we observe that the architecture that consists of stacks based on different basis functions wins over the architecture based on the same stack. It looks like chaining stacks of different nature results in synergistic effects. This is logical as function classes that can be modelled by trend and seasonality stacks have small overlap.

Ensemble size. Figure 3 demonstrates that increasing the ensemble size results in improved performance. Most importantly, according to Figure 3, N-BEATS achieves state-of-the-art performance even if comparatively small ensemble size of 18 models is used. Therefore, computational efficiency of N-BEATS can be traded very effectively for performance and there is no over-reliance of the results on large ensemble size.

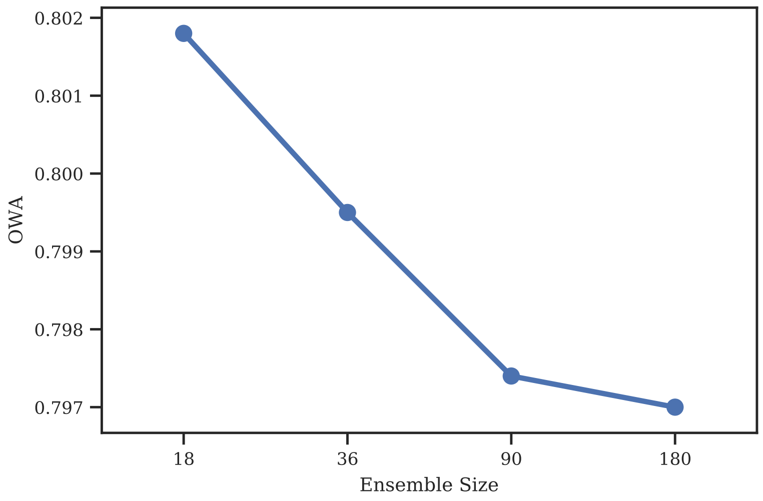


Figure 3: M4 test performance (OWA) as a function of ensemble size, based on N-BEATS-G. This figure shows that N-BEATS loses less than 0.5% in terms of OWA performance even if 10 times smaller ensemble size is used.

Table 7: Performance on the M4 test set, sMAPE. Lower values are better. Red – second best.

	Yearly (23k)	Quarterly (24k)	Monthly (48k)	Others (5k)	Average (100k)
Best pure ML	14.397	11.031	13.973	4.566	12.894
Best statistical	13.366	10.155	13.002	4.682	11.986
Best ML/TS combination	13.528	9.733	12.639	4.118	11.720
DL/TS hybrid, M4 winner	13.176	9.679	12.126	4.014	11.374
N-BEATS-G	13.023	**9.212**	12.048	**3.574**	11.168
N-BEATS-I	12.924	9.287	12.059	3.684	11.174
N-BEATS-I+G	**12.913**	9.213	**12.024**	3.643	**11.135**

Table 8: Performance on the M4 test set, OWA and M4 rank. Lower values are better. Red – second best.

	Yearly (23k)	Quarterly (24k)	Monthly (48k)	Others (5k)	Average (100k)	Rank
Best pure ML	0.859	0.939	0.941	0.991	0.915	23
Best statistical	0.788	0.898	0.905	0.989	0.861	8
Best ML/TS combination	0.799	0.847	0.858	0.914	0.838	2
DL/TS hybrid, M4 winner	0.778	0.847	0.836	0.920	0.821	1
N-BEATS-G	0.765	**0.800**	0.820	**0.822**	0.797	
N-BEATS-I	**0.758**	0.807	0.824	0.849	0.798	
N-BEATS-I+G	**0.758**	**0.800**	**0.819**	0.840	**0.795**	

D DETAILED EMPIRICAL RESULTS

D.1 DETAILED RESULTS: M4 DATASET

Tables 7 and 8 present our key quantitative empirical results showing that the proposed model achieves the state of the art performance on the challenging M4 benchmark. We study the performance of two model configurations: generic (Ours-G) and interpretable (Ours-I), as well as Ours-I+G (ensemble of all models from Ours-G and Ours-I). We compare against 4 representatives from the M4 competition: each best in their respective model class. *Best pure ML* is the submission by B. Trotta, the best entry among the 6 pure ML models. *Best statistical* is the best pure statistical model by N.Z. Legaki and K. Koutsouri. *Best ML/TS combination* is the model by P. Montero-Manso, T. Talagala, R.J. Hyndman and G. Athanasopoulos, second best entry, gradient boosted tree over a few statistical time series models. Finally, *DL/TS hybrid* is the winner of M4 competition designed by S. Smyl.

N-BEATS outperforms all other approaches on all the studied subsets of time series. The average OWA gap between our generic model and the M4 winner ($0.821 - 0.795 = 0.026$) is greater than the gap between the M4 winner and the second entry ($0.838 - 0.821 = 0.017$).

A more granular and detailed statistical analysis of our results on M4 is provided in Table 9. This table first presents the sMAPE for N-BEATS, decomposed by M4 time series sub-type and sampling frequency (upper part). Then (lower part), it shows the *average* sMAPE *difference* between the N-BEATS results and the M4 winner (TS/DL hybrid by S. Smyl), adding the standard error of that difference (in parentheses); bold entries indicate statistical significance at the 99% level based on a two-sided paired t-test.

We note that each cross-section of the M4 dataset into horizon and type may be regarded as an independent mini-dataset. We observe that over those mini-datasets there is a preponderance of statistically significant differences between N-BEATS and Smyl (18 cases out of 31) to the advantage of N-BEATS. This provides evidence that (i) the improvement observed on average in Tables 7 and 8 is statistically significant and consistent over smaller subsets of M4 and (ii) N-BEATS generalizes well over time series of different types and sampling frequencies.

Table 9: Performance decomposition on non-overlapping subsets of the M4 test set and comparison with the Smyl model results.

	Demographic	Finance	Industry	Macro	Micro	Other
sMAPE per M4 series type and sampling frequency						
Yearly	8.931	13.741	16.317	13.327	10.489	13.320
Quarterly	9.219	10.787	8.628	8.576	9.264	6.250
Monthly	4.357	13.353	12.657	12.571	13.627	11.595
Weekly	4.580	3.004	9.258	7.220	10.425	6.183
Daily	6.351	3.467	3.835	2.525	2.299	2.885
Hourly						8.197
Average sMAPE difference vs Smyl model, computed as N-BEATS – Smyl. *Standard error of the mean displayed in parenthesis. Bold entries are significant at the 99% level (2-sided paired t-test).*						
Yearly	**−0.749**	**−0.337**	−0.065	**−0.386**	**−0.168**	−0.157
	(**0.119**)	(0.065)	(0.087)	(**0.085**)	(**0.056**)	(0.140)
Quarterly	**−0.651**	**−0.281**	**−0.328**	**−0.712**	**−0.523**	−0.029
	(**0.085**)	(**0.047**)	(**0.043**)	(**0.060**)	(**0.051**)	(0.083)
Monthly	**−0.185**	**−0.379**	**−0.419**	0.089	**0.338**	−0.279
	(**0.023**)	(**0.034**)	(**0.036**)	(0.039)	(**0.034**)	(0.162)
Weekly	−0.336	**−1.075**	−0.937	−1.627	**−3.029**	−1.193
	(0.270)	(**0.221**)	(1.399)	(0.770)	(**0.378**)	(0.772)
Daily	0.191	**−0.098**	**−0.124**	−0.026	**−0.367**	−0.037
	(0.231)	(**0.018**)	(**0.025**)	(0.057)	(**0.013**)	(0.015)
Hourly						**−1.132**
						(**0.163**)

Table 10: Performance on the M3 test set, Average sMAPE, aggregate over all forecast horizons (Yearly: 1-6, Quarterly: 1-8, Monthly: 1-18, Other: 1-8, Average: 1-18). Lower values are better. Red – second best. †Numbers are computed by us.

	Yearly (645)	Quarterly (756)	Monthly (1428)	Others (174)	Average (3003)
Naïve2	17.88	9.95	16.91	6.30	15.47
ARIMA (B–J automatic)	17.73	10.26	14.81	5.06	14.01
Comb S-H-D	17.07	9.22	14.48	4.56	13.52
ForecastPro	17.14	9.77	13.86	4.60	13.19
Theta	16.90	8.96	13.85	4.41	13.01
DOTM (Fiorucci et al., 2016)	15.94	9.28	13.74	4.58	12.90
EXP (Spiliotis et al., 2019)	16.39	8.98	13.43	5.46	12.71†
LGT (Smyl & Kuber, 2016)	**15.23**	n/a	n/a	4.26	n/a
BaggedETS.BC (Bergmeir et al., 2016)	17.49	9.89	13.74	n/a	n/a
N-BEATS-G	16.2	8.92	13.19	**4.19**	12.47
N-BEATS-I	15.84	9.03	13.15	4.30	12.43
N-BEATS-I+G	15.93	**8.84**	**13.11**	4.24	**12.37**

D.2 Detailed results: M3 Dataset

Results for M3 dataset are provided in Table 10. The performance metric is calculated using the earlier version of sMAPE, defined specifically for the M3 competition:[1]

$$\text{sMAPE} = \frac{200}{H} \sum_{i=1}^{H} \frac{|y_{T+i} - \widehat{y}_{T+i}|}{y_{T+i} + \widehat{y}_{T+i}}. \tag{3}$$

For some of the methods, either average sMAPE was not reported or sMAPE for some of the splits was not reported in their respective publications. Below, we list those cases. BaggedETS.BC (Bergmeir et al., 2016) has not reported numbers on Others. LGT (Smyl & Kuber, 2016) did not report results on Monthly and Quarterly data. According to the authors, the underlying RNN had problems dealing with raw seasonal data, the ETS based pre-processing was not effective and the LGT pre-processing was not computationally feasible given comparatively large number of time series and their comparatively large length (Smyl & Kuber, 2016). Finally, EXP (Spiliotis et al., 2019) reported average performance computed using a different methodology than the default M3 and M4 methodology (source: personal communication with the authors). For the latter method we recomputed the Average sMAPE based on the previously reported Yearly, Quarterly and Monthly splits. To calculate it, we follow the M3, M4 and TOURISM competition methodology and compute the average metric as the average over all time series and over all forecast horizons. Given the performance metric values aggregated over Yearly, Quarterly and Monthly splits, the average can be computed straightforwardly as:

$$\text{sMAPE}_{\text{Average}} = \frac{N_{\text{Year}}}{N_{\text{Tot}}}\text{sMAPE}_{\text{Year}} + \frac{N_{\text{Quart}}}{N_{\text{Tot}}}\text{sMAPE}_{\text{Quart}} + \frac{N_{\text{Month}}}{N_{\text{Tot}}}\text{sMAPE}_{\text{Month}} + \frac{N_{\text{Others}}}{N_{\text{Tot}}}\text{sMAPE}_{\text{Others}}. \tag{4}$$

Here $N_{\text{Tot}} = N_{\text{Year}} + N_{\text{Quart}} + N_{\text{Month}} + N_{\text{Others}}$ and $N_{\text{Year}} = 6 \times 645, N_{\text{Quart}} = 8 \times 756, N_{\text{Month}} = 18 \times 1428, N_{\text{Others}} = 8 \times 174$. It is clear that for each split, its N is the product of its respective number of time series and its largest forecast horizon.

[1]With minor differences compared to the sMAPE definition used for M4. Please refer to Appendix A in (Makridakis & Hibon, 2000) for the mathematical definition.

Table 11: Performance on the TOURISM test set, Average MAPE, aggregate over all forecast horizons (Yearly: 1-4, Quarterly: 1-8, Monthly: 1-24, Average: 1-24). Lower values are better. Red – second best.

	Yearly (518)	Quarterly (427)	Monthly (366)	Average (1311)
Statistical benchmarks (Athanasopoulos et al., 2011)				
SNaïve	23.61	16.46	22.56	21.25
Theta	23.45	16.15	22.11	20.88
ForePro	26.36	15.72	19.91	19.84
ETS	27.68	16.05	21.15	20.88
Damped	28.15	15.56	23.47	22.26
ARIMA	28.03	16.23	21.13	20.96
Kaggle competitors (Athanasopoulos & Hyndman, 2011)				
SaliMali	n/a	14.83	19.64	n/a
LeeCBaker	22.73	15.14	20.19	19.35
Stratometrics	23.15	15.14	20.37	19.52
Robert	n/a	14.96	20.28	n/a
Idalgo	n/a	15.07	20.55	n/a
N-BEATS-G (Ours)	21.67	**14.71**	**19.17**	**18.47**
N-BEATS-I (Ours)	21.55	15.22	19.82	18.97
N-BEATS-I+G (Ours)	**21.44**	14.78	19.29	18.52

D.3 DETAILED RESULTS: TOURISM DATASET

Detailed results for the TOURISM competition dataset are provided in Table 11. The respective Kaggle competition was divided into two parts: (i) Yearly time series forecasting and (ii) Quarterly/Monthly time series forecasting (Athanasopoulos & Hyndman, 2011). Some of the participants chose to take part only in the second part. Therefore, In addition to entries present in Table 1, we report competitors from (Athanasopoulos & Hyndman, 2011) that have missing results in Yearly competition. In particular, *SaliMali* team is the winner of the Quarterly/Monthly time series forecasting competition (Brierley, 2011). Their approach is based on a weighted ensemble of statistical methods. Teams *Robert* and *Idalgo* used unknown approaches. We can see from Table 11 that N-BEATS achieves state-of-the-art performance on all subsets of TOURISM dataset. On average, it is state of the art and it gains 4.2% over the best-known approach *LeeCBaker*, and 11.5% over auto-ARIMA.

The average metrics have not been reported in the original competition results (Athanasopoulos et al., 2011; Athanasopoulos & Hyndman, 2011). Therefore, in Table 11, we present the Average MAPE metric calculated by us based on the previously reported Yearly, Quarterly and Monthly splits. To calculate it, we follow the M4 competition methodology and compute the average metric as the average over all time series and over all forecast horizons. Given the performance metric values aggregated over Yearly, Quarterly and Monthly splits, the average can be computed straightforwardly as:

$$\text{MAPE}_{\text{Average}} = \frac{N_{\text{Year}}}{N_{\text{Tot}}}\text{MAPE}_{\text{Year}} + \frac{N_{\text{Quart}}}{N_{\text{Tot}}}\text{MAPE}_{\text{Quart}} + \frac{N_{\text{Month}}}{N_{\text{Tot}}}\text{MAPE}_{\text{Month}}. \quad (5)$$

Here $N_{\text{Tot}} = N_{\text{Year}} + N_{\text{Quart}} + N_{\text{Month}}$ and $N_{\text{Year}} = 4 \times 518, N_{\text{Quart}} = 8 \times 427, N_{\text{Month}} = 24 \times 366$. It is clear that for each split, its N is the product of its respective number of time series and its largest forecast horizon.

E HYPER-PARAMETER SETTINGS

Table 12 presents the hyperparameter settings used to train models on different subsets of M4, M3 and TOURISM datasets. A brief discussion of field names in the table is warranted.

Subset names **Yly, Qly, Mly, Wly, Dly, Hly, Other** correspond to yearly, quarterly, monthly, weekly, daily, hourly and other frequency subsets defined in the original datasets.

Table 12: Settings of hyperparameters across subsets of M4, M3, TOURISM datasets.

	M4						M3				TOURISM		
	Yly	Qly	Mly	Wly	Dly	Hly	Yly	Qly	Mly	Other	Yly	Qly	Mly
Parameter	N-BEATS-I												
L_H	1.5	1.5	1.5	10	10	10	20	5	5	20	20	10	20
Iterations	15K	15K	15K	5K	5K	5K	50	6K	6K	250	30	500	300
Losses	SMAPE/MAPE/MASE						SMAPE/MAPE/MASE				MAPE		
S-width	2048												
S-blocks	3												
S-block-layers	4												
T-width	256												
T-degree	2												
T-blocks	3												
T-block-layers	4												
Sharing	STACK LEVEL												
Lookback period	$2H,3H,4H,5H,6H,7H$												
Batch	1024												
Parameter	N-BEATS-G												
L_H	1.5	1.5	1.5	10	10	10	20	20	20	10	5	10	20
Iterations	15K	15K	15K	5K	5K	5K	20	250	10K	250	30	100	100
Losses	SMAPE/MAPE/MASE						SMAPE/MAPE/MASE				MAPE		
Width	512												
Blocks	1												
Block-layers	4												
Stacks	30												
Sharing	NO												
Lookback period	$2H,3H,4H,5H,6H,7H$												
Batch	1024												

N-BEATS-I and **N-BEATS-G** correspond to the interpretable and generic model configurations defined in Section 3.3.

E.1 COMMON PARAMETERS

L_H is the coefficient defining the length of training history immediately preceding the last point in the train part of the TS that is used to generate training samples. For example, if for M4 Yearly the forecast horizon is 6 and L_H is 1.5, then we consider $1.5 \cdot 6 = 9$ most recent points in the train dataset for each time series to generate training samples. A training sample from a given TS in M4 Yearly is then generated by choosing one of the most recent 9 points as an anchor. All the points preceding the anchor are used to create the input to N-BEATS, while the points following and including the anchor become training target. Target and history points that fall outside of the time series limits given the anchor position are filled with zeros and masked during the training. We observed that for subsets with large number of time series L_H tends to be smaller and for subsets with smaller number of time series it tends to be larger. For example, in massive Yearly, Monthly, Quarterly subsets of M4 L_H is equal to 1.5; and in moderate to small Weekly, Daily, Hourly subsets of M4 L_H is equal to 10.

Iterations is the number of batches used to train N-BEATS.

Losses is the set of loss functions that is used to build ensemble. We observed on the respective validation sets that for M4 and M3 mixing models trained on a variety of metrics resulted in performance gain. In the case of TOURISM dataset training only on MAPE led to the best validation scores.

Sharing defines whether the coefficients in the fully-connected layers are shared. We observed that the interpretable model works best when weights are shared across stack, while generic model works best when none of the weights are shared.

Lookback period is the length of the history window forming the input to the model (please refer to Figure 1). This is the function of the forecast horizon length, H. In our experiments we mixed models with lookback periods $2H, 3H, 4H, 5H, 6H, 7H$ in one ensemble. As an example, for a forecast horizon length $H = 8$ and a lookback period $7H$, the model's input will consist of the history window of $7 \cdot 8 = 56$ samples.

Batch is the batch size. We used batch size of 1024. We observed that the training was faster with larger batch sizes, however in our setup little gain was observed with batch sizes beyond 1024.

E.2 N-BEATS-I PARAMETERS

S-width is the width of the fully connected layers in the blocks comprising the seasonality stack of the interpretable model (please refer to Figure 1).

S-blocks is the number of blocks comprising the seasonality stack of the interpretable model (please refer to Figure 1).

S-block-layers is the number of fully-connected layers comprising one block in the seasonality stack of the interpretable model (preceding the final fully-connected projection layers forming the backcast/forecast fork, please refer to Figure 1).

T-width is the width of the fully connected layers in the blocks comprising the trend stack of the interpretable model (please refer to Figure 1).

T-degree is the degree p of polynomial in the trend stack of the interpretable model (please refer to equation (1)).

T-blocks is the number of blocks comprising the trend stack of the interpretable model (please refer to Figure 1).

T-block-layers is the number of fully-connected layers comprising one block in the trend stack of the interpretable model (preceding the final fully-connected projection layers forming the backcast/forecast fork, please refer to Figure 1).

E.3 N-BEATS-G PARAMETERS

Width is the width of the fully connected layers in the blocks comprising the stacks of the generic model (please refer to Figure 1).

Blocks is the number of blocks comprising the stack of the generic model (please refer to Figure 1).

Block-layers is the number of fully-connected layers comprising one block in the stack of the generic model (preceding the final fully-connected projection layers forming the backcast/forecast fork, please refer to Figure 1).