Meta-Learning Framework with Applications to Zero-Shot Time-Series Forecasting

Boris N. Oreshkin[1], Dmitri Carpov[1], Nicolas Chapados[1], Yoshua Bengio[2]

[1]Element AI, [2]Mila
boris.oreshkin@gmail.com

Abstract

Can meta-learning discover generic ways of processing time series (TS) from a diverse dataset so as to greatly improve generalization on new TS coming from different datasets? This work provides positive evidence to this using a broad meta-learning framework which we show subsumes many existing meta-learning algorithms. Our theoretical analysis suggests that residual connections act as a meta-learning adaptation mechanism, generating a subset of task-specific parameters based on a given TS input, thus gradually expanding the expressive power of the architecture on-the-fly. The same mechanism is shown via linearization analysis to have the interpretation of a sequential update of the final linear layer. Our empirical results on a wide range of data emphasize the importance of the identified meta-learning mechanisms for successful zero-shot univariate forecasting, suggesting that it is viable to train a neural network on a source TS dataset and deploy it on a different target TS dataset without retraining, resulting in performance that is at least as good as that of state-of-practice univariate forecasting models.

1 Introduction

Time series (TS) forecasting is both a fundamental scientific problem and one of great practical importance. It is central to the actions of intelligent agents: the ability to plan and control as well as to appropriately react to manifestations of complex partially or completely unknown systems often relies on the ability to forecast relevant observations based on past history. Moreover, for most utility-maximizing agents, gains in forecasting accuracy broadly translate into utility gains; as such, improvements in forecasting technology can have wide impacts. Unsurprisingly, forecasting methods have a long history that can be traced back to the very origins of human civilization (Neale 1985), modern science (Gauss 1809) and have consistently attracted considerable research attention (Yule 1927; Walker 1931; Holt 1957; Winters 1960; Engle 1982; Sezer, Gudelek, and Ozbayoglu 2019). The applications of forecasting span a variety of fields, including high-frequency control (e.g. vehicle and robot control (Tang and Salakhutdinov 2019), data center optimization (Gao 2014)), business planning (supply chain management (Leung 1995), workforce and call center management (Chapados et al. 2014;

Ibrahim et al. 2016), as well as such critically important areas as precision agriculture (Rodrigues Jr et al. 2019). In business specifically, improved forecasting translates in better production planning (leading to less waste) and less transportation (reducing CO_2 emissions) (Kahn 2003; Kerkkänen, Korpela, and Huiskonen 2009; Nguyen, Ni, and Rossetti 2010). The progress made in univariate forecasting in the past four decades is well reflected in the results and methods considered in associated competitions over that period (Makridakis et al. 1982, 1993; Makridakis and Hibon 2000; Athanasopoulos et al. 2011; Makridakis, Spiliotis, and Assimakopoulos 2018a). Recently, growing evidence has started to emerge suggesting that machine learning approaches could improve on classical forecasting methods, in contrast to some earlier assessments (Makridakis, Spiliotis, and Assimakopoulos 2018b). For example, the winner of the 2018 M4 competition (Makridakis, Spiliotis, and Assimakopoulos 2018a) was a neural network designed by Smyl (2020).

On the practical side, the deployment of deep neural time-series models is challenged by the cold start problem. Before a *tabula rasa* deep neural network provides a useful forecasting output, it should be trained on a large problem-specific time-series dataset. For early adopters, this often implies data collection efforts, changing data handling practices and even changing the existing IT infrastructures on a large scale. In contrast, advanced statistical models can be deployed with significantly less effort as they estimate their parameters on single time series at a time. In this paper we address the problem of reducing the entry cost of deep neural networks in the industrial practice of TS forecasting. We show that it is viable to train a neural network model on a diversified source dataset and deploy it on a target dataset in a *zero-shot regime*, i.e. without explicit retraining on that target data, resulting in performance that is at least as good as that of advanced statistical models tailored to the target dataset. We would like to clarify that we use the term "zero-shot" in our work in the sense that the number of history samples available for the target time series is so small that it makes training a deep learning model on this time series infeasible.

Addressing this practical problem provides clues to fundamental questions. Can we learn something general about forecasting and transfer this knowledge across datasets? If so, what kind of mechanisms could facilitate this? The ability to learn and transfer representations across tasks via

task adaptation is an advantage of meta-learning (Raghu et al. 2020). We propose here a broad theoretical framework for meta-learning that encompasses several existing meta-learning algorithms. We further show that a recent successful model, N-BEATS (Oreshkin et al. 2020), fits this framework. We identify internal meta-learning adaptation mechanisms that generate new parameters on-the-fly, specific to a given TS, iteratively extending the architecture's expressive power. We empirically confirm that meta-learning mechanisms are key to improving zero-shot TS forecasting performance, and demonstrate results on a wide range of datasets.

1.1 Background

The univariate point forecasting problem in discrete time is formulated given a length-H forecast horizon and a length-T observed series history $[y_1, \ldots, y_T] \in \mathbb{R}^T$. The task is to predict the vector of future values $\mathbf{y} \in \mathbb{R}^H = [y_{T+1}, y_{T+2}, \ldots, y_{T+H}]$. For simplicity, we will later consider a *lookback window* of length $t \leq T$ ending with the last observed value y_T to serve as model input, and denoted $\mathbf{x} \in \mathbb{R}^t = [y_{T-t+1}, \ldots, y_T]$. We denote $\widehat{\mathbf{y}}$ the point forecast of $\mathbf{y}$. Its accuracy can be evaluated with sMAPE, the symmetric mean absolute percentage error (Makridakis, Spiliotis, and Assimakopoulos 2018a),

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

Other quality metrics (e.g. MAPE, MASE, OWA, ND) are possible and are defined in Appendix A.

Meta-learning or *learning-to-learn* (Harlow 1949; Schmidhuber 1987; Bengio, Bengio, and Cloutier 1991) is usually linked to being able to (i) accumulate knowledge across tasks (*i.e. transfer learning*, *multi-task learning*) and (ii) quickly adapt the accumulated knowledge to the new task (*task adaptation*) (Ravi and Larochelle 2016; Bengio et al. 1992).

N-BEATS algorithm has demonstrated outstanding performance on several competition benchmarks (Oreshkin et al. 2020). The model consists of a total of L blocks connected using a doubly residual architecture. Block ℓ has input $\mathbf{x}_\ell$ and produces two outputs: the *backcast* $\widehat{\mathbf{x}}_\ell$ and the *partial forecast* $\widehat{\mathbf{y}}_\ell$. For the first block we define $\mathbf{x}_1 \equiv \mathbf{x}$, where $\mathbf{x}$ is assumed to be the model-level input from now on. We define the k-th fully-connected layer in the ℓ-th block; having RELU non-linearity, weights $\mathbf{W}_k$, bias $\mathbf{b}_k$ and input $\mathbf{h}_{\ell,k-1}$, as $\text{FC}_k(\mathbf{h}_{\ell,k-1}) \equiv \text{RELU}(\mathbf{W}_k\mathbf{h}_{\ell,k-1} + \mathbf{b}_k)$. We focus on the configuration that shares all learnable parameters across blocks. With this notation, one block of N-BEATS is described as:

$$\begin{aligned} \mathbf{h}_{\ell,1} &= \text{FC}_1(\mathbf{x}_\ell), \ \mathbf{h}_{\ell,k} = \text{FC}_k(\mathbf{h}_{\ell,k-1}), \ k = 2 \ldots K; \\ \widehat{\mathbf{x}}_\ell &= \mathbf{Q}\mathbf{h}_{\ell,K}, \qquad \widehat{\mathbf{y}}_\ell = \mathbf{G}\mathbf{h}_{\ell,K}, \end{aligned} \qquad (2)$$

where $\mathbf{Q}$ and $\mathbf{G}$ are linear operators. The N-BEATS parameters included in the FC and linear layers are learned by minimizing a suitable loss function (*e.g.* sMAPE defined in (1)) across multiple TS. Finally, the doubly residual architecture is described by the following recursion (recalling that $\mathbf{x}_1 \equiv \mathbf{x}$):

$$\mathbf{x}_\ell = \mathbf{x}_{\ell-1} - \widehat{\mathbf{x}}_{\ell-1}, \quad \widehat{\mathbf{y}} = \sum_{\ell=1}^{L} \widehat{\mathbf{y}}_\ell. \qquad (3)$$

1.2 Related Work

From a high-level perspective, there are many links with classical TS modeling: a human-specified classical model is typically designed to generalize well on unseen TS, while we propose to automate that process. The classical models include exponential smoothing with and without seasonal effects (Holt 1957, 2004; Winters 1960), multi-trace exponential smoothing approaches, *e.g.* Theta and its variants (Assimakopoulos and Nikolopoulos 2000; Fiorucci et al. 2016; Spiliotis, Assimakopoulos, and Nikolopoulos 2019). Finally, the state space modeling approach encapsulates most of the above in addition to auto-ARIMA and GARCH (Engle 1982; see Hyndman and Khandakar (2008) for an overview). The state-space approach has also been underlying significant amounts of research in the neural TS modeling (Salinas et al. 2019; Wang et al. 2019; Rangapuram et al. 2018). However, those models have not been considered in the zero-shot scenario. In this work we focus on studying the importance of meta-learning for successful zero-shot forecasting. The foundations of meta-learning have been developed by Schmidhuber (1987); Bengio, Bengio, and Cloutier (1991) among others. More recently, meta-learning research has been expanding, mostly outside of the TS forecasting domain (Ravi and Larochelle 2016; Finn, Abbeel, and Levine 2017; Snell, Swersky, and Zemel 2017; Vinyals et al. 2016; Rusu et al. 2019). In the TS domain, meta-learning has manifested itself via neural models trained over a collection of TS (Smyl 2020; Oreshkin et al. 2020) or via a model trained to predict weights combining outputs of several classical forecasting algorithms (Montero-Manso et al. 2020). Successful application of a neural TS forecasting model trained on a source dataset and fine-tuned on the target dataset was demonstrated by Hooshmand and Sharma (2019); Ribeiro et al. (2018) as well as in the context of TS classification by Fawaz et al. (2018). Unlike those, we focus on the zero-shot scenario and address the cold start problem.

1.3 Summary of Contributions

We define a **meta-learning framework** with associated equations, and recast within it many existing meta-learning algorithms. We show that N-BEATS follows the same equations. According to our analysis, its residual connections implement meta-learning inner loop, thereby performing task adaptation without gradient steps at inference time.

We define a novel **zero-shot univariate TS forecasting task** and make its dataset loaders and evaluation code public, including a new large-scale dataset (FRED) with 290k TS.

We empirically show, for the first time, that **deep-learning zero-shot time series forecasting is feasible** and that the meta-learning component is important for zero-shot generalization in univariate TS forecasting.

2 Meta-Learning Framework

A meta-learning procedure can generally be viewed at two levels: the *inner loop* and the *outer loop*. The inner training loop operates within an individual "meta-example" or task $\mathcal{T}$ (fast learning loop improving over current $\mathcal{T}$) and the outer loop operates across tasks (slow learning loop). A task $\mathcal{T}$

includes task training data $\mathcal{D}_{\mathcal{T}}^{tr}$ and task validation data $\mathcal{D}_{\mathcal{T}}^{val}$, both optionally involving inputs, targets and a task-specific loss: $\mathcal{D}_{\mathcal{T}}^{tr} = \{\mathcal{X}_{\mathcal{T}}^{tr}, \mathcal{Y}_{\mathcal{T}}^{tr}, \mathcal{L}_{\mathcal{T}}\}$, $\mathcal{D}_{\mathcal{T}}^{val} = \{\mathcal{X}_{\mathcal{T}}^{val}, \mathcal{Y}_{\mathcal{T}}^{val}, \mathcal{L}_{\mathcal{T}}\}$. Accordingly, a meta-learning set-up can be defined by assuming a distribution $p(\mathcal{T})$ over tasks, a predictor $\mathcal{P}_{\theta,\mathbf{w}}$ and a meta-learner with meta-parameters φ. We allow a subset of predictor's parameters denoted $\mathbf{w}$ to belong to meta-parameters φ and hence not to be task adaptive. The objective is to design a meta-learner that can generalize well on a new task by appropriately choosing the predictor's task adaptive parameters θ after observing $\mathcal{D}_{\mathcal{T}}^{tr}$. The meta-learner is trained to do so by being exposed to many tasks in a training dataset $\{\mathcal{T}_i^{train}\}$ sampled from $p(\mathcal{T})$. For each training task $\mathcal{T}_i^{train}$, the meta-learner is requested to produce the solution to the task in the form of $\mathcal{P}_{\theta,\mathbf{w}} : \mathcal{X}_{\mathcal{T}_i}^{val} \mapsto \widehat{\mathcal{Y}}_{\mathcal{T}_i}^{val}$ conditioned on $\mathcal{D}_{\mathcal{T}_i}^{tr}$. The meta-parameters φ are updated in the outer meta-learning loop so as to obtain good generalization in the inner loop, *i.e.*, by minimizing the expected validation loss $\mathbb{E}_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(\widehat{\mathcal{Y}}_{\mathcal{T}_i}^{val}, \mathcal{Y}_{\mathcal{T}_i}^{val})$ mapping the ground truth and estimated outputs into the value that quantifies the generalization performance across tasks. Training on multiple tasks enables the meta-learner to produce solutions $\mathcal{P}_{\theta,\mathbf{w}}$ that generalize well on a set of unseen tasks $\{\mathcal{T}_i^{test}\}$ sampled from $p(\mathcal{T})$.

Consequently, the meta-learning procedure has three distinct ingredients: (i) meta-parameters $\varphi = (\mathbf{t}_0, \mathbf{w}, \mathbf{u})$, (ii) initialization function $\mathcal{I}_{\mathbf{t}_0}$ and (iii) update function $\mathcal{U}_{\mathbf{u}}$. The **meta-learner's meta-parameters** φ include the meta-parameters of the meta-initialization function, $\mathbf{t}_0$, the meta-parameters of the predictor shared across tasks, $\mathbf{w}$, and the meta-parameters of the update function, $\mathbf{u}$. The **meta-initialization function** $\mathcal{I}_{\mathbf{t}_0}(\mathcal{D}_{\mathcal{T}_i}^{tr}, \mathbf{c}_{\mathcal{T}_i})$ defines the initial values of parameters θ for a given task $\mathcal{T}_i$ based on its meta-initialization parameters $\mathbf{t}_0$, task training dataset $\mathcal{D}_{\mathcal{T}_i}^{tr}$ and task meta-data $\mathbf{c}_{\mathcal{T}_i}$. Task meta-data may have, for example, a form of task ID or a textual task description. **The update function** $\mathcal{U}_{\mathbf{u}}(\theta_{\ell-1}, \mathcal{D}_{\mathcal{T}_i}^{tr})$ is parameterized with update meta-parameters $\mathbf{u}$. It defines an iterated update to predictor parameters θ at iteration ℓ based on their previous value and the task training set $\mathcal{D}_{\mathcal{T}_i}^{tr}$. The initialization and update functions produce a sequence of predictor parameters, which we compactly write as $\theta_{0:\ell} \equiv \{\theta_0, \ldots, \theta_{\ell-1}, \theta_\ell\}$. We let the final predictor be a function of the whole sequence of parameters, written compactly as $\mathcal{P}_{\theta_{0:\ell},\mathbf{w}}$. One implementation of such general function could be a Bayesian ensemble or a weighted sum, for example: $\mathcal{P}_{\theta_{0:\ell},\mathbf{w}}(\cdot) = \sum_{j=0}^{\ell} \omega_j \mathcal{P}_{\theta_j,\mathbf{w}}(\cdot)$. If we set $\omega_j = 1$ iff $j = \ell$ and 0 otherwise, then we get the more common situation $\mathcal{P}_{\theta_{0:\ell},\mathbf{w}}(\cdot) \equiv \mathcal{P}_{\theta_\ell,\mathbf{w}}(\cdot)$. This meta-learning framework is succinctly described by the following set of equations:

$$\text{Parameters: } \theta; \quad \text{Meta-parameters: } \varphi = (\mathbf{t}_0, \mathbf{w}, \mathbf{u})$$

$$\text{Inner Loop: } \theta_0 \leftarrow \mathcal{I}_{\mathbf{t}_0}(\mathcal{D}_{\mathcal{T}_i}^{tr}, \mathbf{c}_{\mathcal{T}_i})$$
$$\theta_\ell \leftarrow \mathcal{U}_{\mathbf{u}}(\theta_{\ell-1}, \mathcal{D}_{\mathcal{T}_i}^{tr}), \; \forall \ell > 0 \quad (4)$$

$$\text{Prediction at } \mathbf{x} : \mathcal{P}_{\theta_{0:\ell},\mathbf{w}}(\mathbf{x})$$

$$\text{Outer Loop: } \varphi \leftarrow \varphi - \eta \nabla_\varphi \mathcal{L}_{\mathcal{T}_i}[\mathcal{P}_{\theta_{0:\ell},\mathbf{w}}(\mathcal{X}_{\mathcal{T}_i}^{val}), \mathcal{Y}_{\mathcal{T}_i}^{val}]. \quad (5)$$

2.1 Meta-Learning and Time-Teries Forecasting

In the previous section we laid out a unifying framework for meta-learning. How is it connected to the TS forecasting task? We believe that this question is best answered by answering questions "why the classical statistical TS forecasting models such as ARIMA and ETS are not doing meta-learning?" and "what does the meta-learning component offer when it is part of a forecasting algorithm?". The first question can be answered by considering the fact that the classical statistical models produce a forecast by estimating their parameters from the history of the target time series using a predefined fixed set of rules, for example, given a model selection and the maximum likelihood parameter estimator for it. Therefore, in terms of our meta-learning framework, a classical statistical model executes only the inner loop (model parameter estimation) encapsulated in equation (4). The outer loop in this case is irrelevant — a human analyst defines what equation (4) is doing, based on experience (for example, "for most slow varying time-series with trend, no seasonality and white residuals, ETS with Gaussian maximum likelihood estimator will probably work well"). The second question can be answered considering that meta-learning based forecasting algorithm replaces the predefined fixed set of rules for model parameter estimation with a learnable parameter estimation strategy. The learnable parameter estimation strategy is trained using outer loop equation (5) by adjusting the strategy such that it is able to produce parameter estimates that generalize well over multiple TS. It is assumed that there exists a dataset that is representative of the forecasting tasks that will be handled at inference time. Thus the main advantage of meta-learning based forecasting approaches is that they enable learning a data-driven parameter estimator that can be optimized for a particular set of forecasting tasks and forecasting models. On top of that, a meta-learning approach allows for a general learnable predictor in equation (4) that can be optimized for a given forecasting task. So both predictor (model) and its parameter estimator can be jointly learned for a forecasting task represented by available data. Empirically, we show that this elegant theoretical concept works effectively across multiple datasets and across multiple forecasting tasks (e.g. forecasting yearly, monthly or hourly TS) and even across very loosely related tasks (for example, forecasting hourly electricity demand after training on a monthly economic data after appropriate time scale normalization).

2.2 Expressing Existing Meta-Learning Algorithms in the Proposed Framework

To further illustrate the generality of the proposed framework, we next show how to cast existing meta-learning algorithms within it, before turning to N-BEATS.

MAML and related approaches (Finn, Abbeel, and Levine 2017; Li et al. 2017; Raghu et al. 2020) can be derived from (4) and (5) by (i) setting $\mathcal{I}$ to be the identity map that copies $\mathbf{t}_0$ into θ, (ii) setting $\mathcal{U}$ to be the SGD gradient update: $\mathcal{U}_{\mathbf{u}}(\theta, \mathcal{D}_{\mathcal{T}_i}^{tr}) = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\mathcal{P}_{\theta,\mathbf{w}}(\mathcal{X}_{\mathcal{T}_i}^{tr}), \mathcal{Y}_{\mathcal{T}_i}^{tr})$, where $\mathbf{u} = \{\alpha\}$ and by (iii) setting the predictor's meta-parameters to the empty set $\mathbf{w} = \emptyset$. Equation (5) applies with no modifications. **MT-net** (Lee and Choi 2018) is a variant of MAML in

which the predictor's meta-parameter set $\mathbf{w}$ is not empty. The part of the predictor parameterized with $\mathbf{w}$ is meta-learned across tasks and is fixed during task adaptation.

Optimization as a model for few-shot learning (Ravi and Larochelle 2016) can be derived from (4) and (5) via the following steps (in addition to those of MAML). First, set the update function $\mathcal{U}_{\mathbf{u}}$ to the update equation of an LSTM-like cell of the form (ℓ is the LSTM update step index) $\theta_\ell \leftarrow f_\ell \theta_{\ell-1} + \alpha_\ell \nabla_{\theta_{\ell-1}} \mathcal{L}_{\mathcal{T}_i}(\mathcal{P}_{\theta_{\ell-1},\mathbf{w}}(\mathcal{X}^{tr}_{\mathcal{T}_i}), \mathcal{Y}^{tr}_{\mathcal{T}_i})$. Second, set f_ℓ to be the LSTM forget gate value (Ravi and Larochelle 2016): $f_\ell = \sigma(\mathbf{W}_F[\nabla_\theta \mathcal{L}_{\mathcal{T}_i}, \mathcal{L}_{\mathcal{T}_i}, \theta_{\ell-1}, f_{\ell-1}] + \mathbf{b}_F)$ and α_ℓ to be the LSTM input gate value: $\alpha_\ell = \sigma(\mathbf{W}_\alpha[\nabla_\theta \mathcal{L}_{\mathcal{T}_i}, \mathcal{L}_{\mathcal{T}_i}, \theta_{\ell-1}, \alpha_{\ell-1}] + \mathbf{b}_\alpha)$. Here σ is a sigmoid non-linearity. Finally, include all the LSTM parameters into the set of update meta-parameters: $\mathbf{u} = \{\mathbf{W}_F, \mathbf{b}_F, \mathbf{W}_\alpha, \mathbf{b}_\alpha\}$.

Prototypical Networks (PNs) (Snell, Swersky, and Zemel 2017). Most metric-based meta-learning approaches, including PNs, rely on comparing embeddings of the task training set with those of the validation set. Therefore, it is convenient to consider a composite predictor consisting of the *embedding* function, $\mathcal{E}_{\mathbf{w}}$, and the *comparison* function, $\mathcal{C}_\theta$, $\mathcal{P}_{\theta,\mathbf{w}}(\cdot) = \mathcal{C}_\theta \circ \mathcal{E}_{\mathbf{w}}(\cdot)$. PNs can be derived from (4) and (5) by considering a K-shot image classification task, convolutional network $\mathcal{E}_{\mathbf{w}}$ shared across tasks and class prototypes $\mathbf{p}_k = \frac{1}{K}\sum_{j:\mathcal{Y}^{tr}_j=k}\mathcal{E}_{\mathbf{w}}(\mathcal{X}^{tr}_j)$ included in $\theta = \{\mathbf{p}_k\}_{\forall k}$. Initialization function $\mathcal{I}_{\mathbf{t}_0}$ with $\mathbf{t}_0 = \emptyset$ simply sets θ to the values of prototypes. $\mathcal{U}_{\mathbf{u}}$ is an identity map with $\mathbf{u} = \emptyset$ and $\mathcal{C}_\theta$ is as a softmax classifier:

$$\mathcal{Y}^{val}_{\mathcal{T}_i} = \arg\max_k \text{softmax}(-d(\mathcal{E}_{\mathbf{w}}(\mathcal{X}^{val}_{\mathcal{T}_i}), \mathbf{p}_k)). \qquad (6)$$

Here $d(\cdot, \cdot)$ is a similarity measure and the softmax is normalized w.r.t. all $\mathbf{p}_k$. Finally, define the loss $\mathcal{L}_{\mathcal{T}_i}$ in (5) as the cross-entropy of the softmax classifier described in (6). Interestingly, $\theta = \{\mathbf{p}_k\}_{\forall k}$ are nothing else than the dynamically generated weights of the final linear layer fed into the softmax, which is especially apparent when $d(\mathbf{a}, \mathbf{b}) = -\mathbf{a} \cdot \mathbf{b}$. The fact that in the prototypical network scenario only the final linear layer weights are dynamically generated based on the task training set resonates very well with the most recent study of MAML (Raghu et al. 2020). It has been shown that most of the MAML's gain can be recovered by only adapting the weights of the final linear layer in the inner loop.

In this section, we illustrated that four distinct meta-learning algorithms from two broad categories (optimization- and metric-based) can be derived from our equations (4) and (5). This confirms that our meta-learning framework is general and it can represent existing meta-learning algorithms. The analysis of three additional existing meta-learning algorithms is presented in Appendix C.

3 N-BEATS as a Meta-learning Algorithm

Let us now focus on the analysis of N-BEATS described by equations (2), (3). We first introduce the following notation: $f : \mathbf{x}_\ell \mapsto \mathbf{h}_{\ell,4}$; $g : \mathbf{h}_{\ell,4} \mapsto \widehat{\mathbf{y}}_\ell$; $q : \mathbf{h}_{\ell,4} \mapsto \widehat{\mathbf{x}}_\ell$. In the original equations, g and q are linear and hence can be represented by equivalent matrices $\mathbf{G}$ and $\mathbf{Q}$. In the following, we keep the notation general as much as possible, transitioning to the linear case only when needed. Then, given the network input,

$\mathbf{x}$ ($\mathbf{x}_1 \equiv \mathbf{x}$), and noting that $\widehat{\mathbf{x}}_{\ell-1} = q \circ f(\mathbf{x}_{\ell-1})$ we can write N-BEATS as follows:

$$\widehat{\mathbf{y}} = g \circ f(\mathbf{x}) + \sum_{\ell>1} g \circ f(\mathbf{x}_{\ell-1} - q \circ f(\mathbf{x}_{\ell-1})). \qquad (7)$$

N-BEATS is now derived from the meta-learning framework of Sec. 2 using two observations: (i) each application of $g \circ f$ in (7) is a predictor and (ii) each block of N-BEATS is the iteration of the inner meta-learning loop. More concretely, we have that $\mathcal{P}_{\theta,\mathbf{w}}(\cdot) = g_{\mathbf{w}_g} \circ f_{\mathbf{w}_f,\theta}(\cdot)$. Here $\mathbf{w}_g$ and $\mathbf{w}_f$ are parameters of functions g and f, included in $\mathbf{w} = (\mathbf{w}_g, \mathbf{w}_f)$ and learned across tasks in the outer loop. The task-specific parameters θ consist of the sequence of input shift vectors, $\theta \equiv \{\mu_\ell\}_{\ell=0}^L$, defined such that the ℓ-th block input can be written as $\mathbf{x}_\ell = \mathbf{x} - \mu_{\ell-1}$. This yields a recursive expression for the predictor's task-specific parameters of the form $\mu_\ell \leftarrow \mu_{\ell-1} + \widehat{\mathbf{x}}_\ell$, $\mu_0 \equiv \mathbf{0}$, obtained by recursively unrolling eq. (3). These yield the following initialization and update functions: $\mathcal{I}_{\mathbf{t}_0}$ with $\mathbf{t}_0 = \emptyset$ sets μ_0 to zero; $\mathcal{U}_{\mathbf{u}}$, with $\mathbf{u} = (\mathbf{w}_q, \mathbf{w}_f)$ generates a next parameter update based on $\widehat{\mathbf{x}}_\ell$:

$$\mu_\ell \leftarrow \mathcal{U}_{\mathbf{u}}(\mu_{\ell-1}, \mathcal{D}^{tr}_{\mathcal{T}_i}) \equiv \mu_{\ell-1} + q_{\mathbf{w}_q} \circ f_{\mathbf{w}_f}(\mathbf{x} - \mu_{\ell-1}).$$

Interestingly, (i) meta-parameters $\mathbf{w}_f$ are shared between the predictor and the update function and (ii) the task training set is limited to the network input, $\mathcal{D}^{tr}_{\mathcal{T}_i} \equiv \{\mathbf{x}\}$. Note that the latter makes sense because the data are complete time series, with the inputs $\mathbf{x}$ having the same form of internal dependencies as the forecasting targets $\mathbf{y}$. Hence, observing $\mathbf{x}$ is enough to infer how to predict $\mathbf{y}$ from $\mathbf{x}$ in a way that is similar to how different parts of $\mathbf{x}$ are related to each other.

Finally, according to (7), predictor outputs corresponding to the values of parameters θ learned at every iteration of the inner loop are combined in the final output. This corresponds to choosing a predictor of the form $\mathcal{P}_{\mu_{0:L},\mathbf{w}}(\cdot) = \sum_{j=0}^L \omega_j \mathcal{P}_{\mu_j,\mathbf{w}}(\cdot), \omega_j = 1, \forall j$ in (5). The outer learning loop (5) describes the N-BEATS training procedure across tasks (TS) with no modification.

It is clear that the final output of the architecture depends on the sequence $\mu_{0:L}$. Even if predictor parameters $\mathbf{w}_g$, $\mathbf{w}_f$ are shared across blocks and fixed, the behaviour of $\mathcal{P}_{\mu_{0:L},\mathbf{w}}(\cdot) = g_{\mathbf{w}_g} \circ f_{\mathbf{w}_f,\mu_{0:L}}(\cdot)$ is governed by an extended space of parameters $(\mathbf{w}, \mu_1, \mu_2, \ldots)$. Therefore, the expressive power of the architecture can be expected to grow with the growing number of blocks, in proportion to the growth of the space spanned by $\mu_{0:L}$, even if $\mathbf{w}_g$, $\mathbf{w}_f$ are shared across blocks. Thus, it is reasonable to expect that the addition of identical blocks will improve generalization performance because of the increase in expressive power.

3.1 Linear Approximation Analysis

Next, we go a level deeper in the analysis to uncover more intricate task adaptation processes. Using linear approximation analysis, we express N-BEATS' meta-learning operation in terms of the adaptation of the internal weights of the network based on the task input data. In particular, assuming small $\widehat{\mathbf{x}}_\ell$, (7) can be approximated using the first order Taylor series expansion in the vicinity of $\mathbf{x}_{\ell-1}$:

$$\widehat{\mathbf{y}} = g \circ f(\mathbf{x}) + \sum_{\ell>1}[g - \mathbf{J}_{g \circ f}(\mathbf{x}_{\ell-1})q] \circ f(\mathbf{x}_{\ell-1})$$
$$+ o(\|q \circ f(\mathbf{x}_{\ell-1})\|).$$

Here $\mathbf{J}_{g\circ f}(\mathbf{x}_{\ell-1}) = \mathbf{J}_g(f(\mathbf{x}_{\ell-1}))\mathbf{J}_f(\mathbf{x}_{\ell-1})$ is the Jacobian of $g \circ f$. We now consider linear g and q, as mentioned earlier, in which case g and q are represented by two matrices of appropriate dimensionality, $\mathbf{G}$ and $\mathbf{Q}$; and $\mathbf{J}_g(f(\mathbf{x}_{\ell-1})) = \mathbf{G}$. Thus, the above expression can be simplified as:

$$\widehat{\mathbf{y}} = \mathbf{G}f(\mathbf{x}) + \sum_{\ell>1}\mathbf{G}[\mathbf{I} - \mathbf{J}_f(\mathbf{x}_{\ell-1})\mathbf{Q}]f(\mathbf{x}_{\ell-1}) + o(\|\mathbf{Q}f(\mathbf{x}_{\ell-1})\|).$$

Continuously applying the linear approximation $f(\mathbf{x}_\ell) = [\mathbf{I} - \mathbf{J}_f(\mathbf{x}_{\ell-1})\mathbf{Q}]f(\mathbf{x}_{\ell-1}) + o(\|\mathbf{Q}f(\mathbf{x}_{\ell-1})\|)$ until we reach $\ell = 1$ and recalling that $\mathbf{x}_1 \equiv \mathbf{x}$ we arrive at the following:

$$\widehat{\mathbf{y}} = \sum_{\ell>0}\mathbf{G}\left[\prod_{k=1}^{\ell-1}[\mathbf{I} - \mathbf{J}_f(\mathbf{x}_{\ell-k})\mathbf{Q}]\right]f(\mathbf{x}) + o(\|\mathbf{Q}f(\mathbf{x}_\ell)\|). \quad (8)$$

Note that $\mathbf{G}\left(\prod_{k=1}^{\ell-1}[\mathbf{I} - \mathbf{J}_f(\mathbf{x}_{\ell-k})\mathbf{Q}]\right)$ can be written in the iterative update form. Consider $\mathbf{G}'_1 = \mathbf{G}$, then the update equation for $\mathbf{G}'$ can be written as $\mathbf{G}'_\ell = \mathbf{G}'_{\ell-1}[\mathbf{I} - \mathbf{J}_f(\mathbf{x}_{\ell-1})\mathbf{Q}], \ \forall \ell > 1$ and (8) becomes:

$$\widehat{\mathbf{y}} = \sum_{\ell>0}\mathbf{G}'_\ell f(\mathbf{x}) + o(\|\mathbf{Q}f(\mathbf{x}_\ell)\|). \quad (9)$$

Let us now discuss how (9) can be used to re-interpret N-BEATS as an instance of the meta-learning framework (4) and (5). The predictor can now be represented in a decoupled form $\mathcal{P}_{\theta,\mathbf{w}}(\cdot) = g_\theta \circ f_{\mathbf{w}_f}(\cdot)$. Thus task adaptation is clearly confined in the decision function, g_θ, whereas the embedding function $f_{\mathbf{w}_f}$ only relies on fixed meta-parameters $\mathbf{w}_f$. The adaptive parameters θ include the sequence of projection matrices $\{\mathbf{G}'_\ell\}$. The meta-initialization function $\mathcal{I}_{\mathbf{t}_0}$ is parameterized with $\mathbf{t}_0 \equiv \mathbf{G}$ and it simply sets $\mathbf{G}'_1 \leftarrow \mathbf{t}_0$. The main ingredient of the update function $\mathcal{U}_\mathbf{u}$ is $\mathbf{Q}f_{\mathbf{w}_f}(\cdot)$, parameterized as before with $\mathbf{u} = (\mathbf{Q}, \mathbf{w}_f)$. The update function now consists of two equations:

$$\begin{aligned} \mathbf{G}'_\ell &\leftarrow \mathbf{G}'_{\ell-1}[\mathbf{I} - \mathbf{J}_f(\mathbf{x} - \mu_{\ell-1})\mathbf{Q}], \quad \forall \ell > 1, \\ \mu_\ell &\leftarrow \mu_{\ell-1} + \mathbf{Q}f_{\mathbf{w}_f}(\mathbf{x} - \mu_{\ell-1}), \quad \mu_0 = \mathbf{0}. \end{aligned} \quad (10)$$

The first order analysis results (9) and (10) suggest that under certain circumstances, the block-by-block manipulation of the input sequence apparent in (7) is equivalent to producing an iterative update of predictor's final linear layer weights apparent in (10), with the block input being set to the same fixed value. This is very similar to the final linear layer update behaviour identified in other meta-learning algorithms: in LEO it is present by design (Rusu et al. 2019), in MAML it was identified by Raghu et al. (2020), and in PNs it follows from the results of our analysis in Section 2.2.

3.2 The Role of Q

It is hard to study the form of $\mathbf{Q}$ learned from the data in general. However, equipped with the results of the linear approximation analysis presented in Section 3.1, we can study the case of a two-block network, assuming that the L^2 norm loss between $\mathbf{y}$ and $\widehat{\mathbf{y}}$ is used to train the network. If, in addition, the dataset consists of the set of N pairs $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}$ the dataset-wise loss $\mathcal{L}$ has the following expression:

$$\mathcal{L} = \sum_i \left\|\mathbf{y}^i - 2\mathbf{G}f(\mathbf{x}^i) + \mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i) + o(\|\mathbf{Q}f(\mathbf{x}^i))\|)\right\|^2.$$

Introducing $\Delta\mathbf{y}^i = \mathbf{y}^i - 2\mathbf{G}f(\mathbf{x}^i)$, the error between the *default* forecast $2\mathbf{G}f(\mathbf{x}^i)$ and the ground truth $\mathbf{y}^i$, and expanding the L^2 norm we obtain the following:

$$\begin{aligned} \mathcal{L} = \sum_i &\Delta\mathbf{y}^{i\mathsf{T}}\Delta\mathbf{y}^i + 2\Delta\mathbf{y}^{i\mathsf{T}}\mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i) \\ &+ f(\mathbf{x}^i)^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{J}_{g\circ f}^\mathsf{T}(\mathbf{x}^i)\mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i) + o(\|\mathbf{Q}f(\mathbf{x}^i))\|). \end{aligned}$$

Now, assuming that the rest of the parameters of the network are fixed, we have the derivative with respect to $\mathbf{Q}$ using matrix calculus (Petersen and Pedersen 2012):

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\mathbf{Q}} = \sum_i &2\mathbf{J}_{g\circ f}^\mathsf{T}(\mathbf{x}^i)\Delta\mathbf{y}^i f(\mathbf{x}^i)^\mathsf{T} \\ &+ 2\mathbf{J}_{g\circ f}^\mathsf{T}(\mathbf{x}^i)\mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i)f(\mathbf{x}^i)^\mathsf{T} + o(\|\mathbf{Q}f(\mathbf{x}^i))\|). \end{aligned}$$

Using the above expression we conclude that the first-order approximation of optimal $\mathbf{Q}$ satisfies the following equation:

$$\sum_i \mathbf{J}_{g\circ f}^\mathsf{T}(\mathbf{x}^i)\Delta\mathbf{y}^i f(\mathbf{x}^i)^\mathsf{T} = -\sum_i \mathbf{J}_{g\circ f}^\mathsf{T}(\mathbf{x}^i)\mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i)f(\mathbf{x}^i)^\mathsf{T}.$$

Although this does not help to find a closed form solution for $\mathbf{Q}$, it does provide a quite obvious intuition: the LHS and the RHS are equal when the correction term created by the second block, $\mathbf{J}_{g\circ f}(\mathbf{x}^i)\mathbf{Q}f(\mathbf{x}^i)$, tends to compensate the default forecast error, $\Delta\mathbf{y}^i$. Therefore, $\mathbf{Q}$ satisfying the equation will tend to drive the update to $\mathbf{G}$ in (10) in such a way that on average the projection of $f(\mathbf{x})$ over the update $\mathbf{J}_{g\circ f}(\mathbf{x})\mathbf{Q}$ to matrix $\mathbf{G}$ will tend to compensate the error $\Delta\mathbf{y}$ made by forecasting $\mathbf{y}$ using $\mathbf{G}$ based on meta-initialization.

3.3 Factors Enabling Meta-learning

Let us now analyze the factors that enable the meta-learning inner loop obvious in (10). First, meta-learning regime is not viable without having multiple blocks connected via the residual connection (feedback loop): $\mathbf{x}_\ell = \mathbf{x}_{\ell-1} - q \circ f(\mathbf{x}_{\ell-1})$. Second, the meta-learning inner loop is not viable when f is linear: the update of $\mathbf{G}$ is extracted from the curvature of f at the point dictated by the input $\mathbf{x}$ and the sequence of shifts $\mu_{0:L}$. Indeed, suppose f is linear, and denote it by linear operator $\mathbf{F}$. The Jacobian $\mathbf{J}_f(\mathbf{x}_{\ell-1})$ becomes a constant, $\mathbf{F}$. Equation (8) simplifies as (note that for linear f, (8) is exact):

$$\widehat{\mathbf{y}} = \sum_{\ell>0}\mathbf{G}[\mathbf{I} - \mathbf{FQ}]^{\ell-1}\mathbf{Fx}.$$

Therefore, $\mathbf{G}\sum_{\ell>0}[\mathbf{I} - \mathbf{FQ}]^{\ell-1}$ may be replaced with an equivalent $\mathbf{G}'$ that is not data adaptive. Interestingly, $\sum_{\ell>0}[\mathbf{I} - \mathbf{FQ}]^{\ell-1}$ happens to be a truncated Neumann series. Denoting Moore-Penrose pseudo-inverse as $[\cdot]^+$, assuming boundedness of $\mathbf{FQ}$ and completing the series, $\sum_{\ell=0}^\infty[\mathbf{I} - \mathbf{FQ}]^\ell$, results in $\widehat{\mathbf{y}} = \mathbf{G}[\mathbf{FQ}]^+\mathbf{Fx}$. Therefore, under certain conditions, the N-BEATS architecture with linear f and infinite number of blocks can be interpreted as a linear predictor of a signal in colored noise. Here the $[\mathbf{FQ}]^+$ part cleans the intermediate space created by projection $\mathbf{F}$ from the components that are undesired for forecasting and $\mathbf{G}$ creates the forecast based on the initial projection $\mathbf{Fx}$ after it is "sanitized" by $[\mathbf{FQ}]^+$.

In this section we established that N-BEATS is an instance of a meta-learning algorithm described by equations (4) and

M4, sMAPE		M3, sMAPE[‡]		TOURISM, MAPE		ELECTR / TRAFF, ND		FRED, sMAPE	
Pure ML	12.89	Comb	13.52	ETS	20.88	MatFact	0.16 / 0.20	ETS	14.16
Best STAT	11.99	ForePro	13.19	Theta	20.88	DeepAR	0.07 / 0.17	Naïve	12.79
ProLogistica	11.85	Theta	13.01	ForePro	19.84	DeepState	0.08 / 0.17	SES	12.70
Best ML/TS	11.72	DOTM	12.90	Strato	19.52	Theta	0.08 / 0.18	Theta	12.20
DL/TS hybrid	11.37	EXP	12.71	LCBaker	19.35	ARIMA	0.07 / 0.15	ARIMA	12.15
N-BEATS	11.14		12.37		18.52		0.07 / 0.11		11.49
DeepAR*	12.25		12.67		19.27		0.09 / 0.19		n/a
DeepAR-M4*	n/a		14.76		24.79		0.15 / 0.36		n/a
N-BEATS-M4	n/a		12.44		18.82		0.09 / 0.15		11.60
N-BEATS-FR	11.70		12.69		19.94	†	0.09 / 0.26		n/a

Table 1: Dataset-specific metrics aggregated over each dataset; lower values are better. The bottom three rows represent the zero-shot transfer setup, indicating respectively the core algorithm (DeepAR or N-BEATS) and the source dataset (M4 or FR(ED)). All other model names are explained in Appendix F. †N-BEATS trained on double upsampled monthly data, see Appendix D. ‡M3/M4 sMAPE definitions differ. *DeepAR trained by us using GluonTS.

(5). We showed that each block of N-BEATS is an inner meta-learning loop that generates additional shift parameters specific to the input time series. Therefore, the expressive power of the architecture is expected to grow with each additional block, even if all blocks share their parameters. We used linear approximation analysis to show that the input shift in a block is equivalent to the update of the block's final linear layer weights under certain conditions. The key role in this process seems to be encapsulated in the non-linearity of f and in residual connections.

4 Empirical Results

We evaluate performance on a number of datasets representing a diverse set of univariate time series. For each of them, we evaluate the base N-BEATS performance compared against the best-published approaches. We also evaluate zero-shot transfer from several source datasets, as explained next.

Base datasets. M4 (M4 Team 2018), contains 100k TS representing demographic, finance, industry, macro and micro indicators. Sampling frequencies include yearly, quarterly, monthly, weekly, daily and hourly. **M3** (Makridakis and Hibon 2000) contains 3003 TS from domains and sampling frequencies similar to M4. **FRED** is a dataset introduced in this paper containing 290k US and international economic TS from 89 sources, a subset of the data published by the Federal Reserve Bank of St. Louis (Federal Reserve 2019). **TOURISM** (Athanasopoulos et al. 2011) includes monthly, quarterly and yearly series of indicators related to tourism activities. **ELECTRICITY** (Dua and Graff 2017; Yu, Rao, and Dhillon 2016) represents the hourly electricity usage of 370 customers. **TRAFFIC** (Dua and Graff 2017; Yu, Rao, and Dhillon 2016) tracks hourly occupancy rates of 963 lanes in the Bay Area freeways. Additional details for all datasets appear in Appendix E.

Zero-shot TS forecasting task definition. One of the base datasets, a *source* dataset, is used to train a machine learning model. The trained model then forecasts a TS in a *target* dataset. The source and the target datasets are distinct: they do not contain TS whose values are linear transformations of each other. The forecasted TS is split into two non-overlapping pieces: the history, and the test. The history is used as model input and the test is used to compute the forecast error metric. We use the history and the test splits for the base datasets consistent with their original publication, unless explicitly stated otherwise. To produce forecasts, the model is allowed to access the TS in the target dataset on a *one-at-a-time* basis. This is to avoid having the model implicitly learn/adapt based on any information contained in the target dataset other than the history of the forecasted TS. If any adjustments of model parameters or hyperparameters are necessary, they are allowed *exclusively* using the history of the forecasted TS.

Training setup. DeepAR (Salinas et al. 2019) is trained using GluonTS implementation from its authors (Alexandrov et al. 2019). N-BEATS is trained following the original training setup of Oreshkin et al. (2020). Both N-BEATS and DeepAR are trained with scaling/descaling the architecture input/output by dividing/multiplying all input/output values by the max value of the input window computed per target time-series. This does not affect the accuracy of the models in the usual train/test scenario. In the zero-shot regime, this operation is intended to prevent catastrophic failure when the scale of the target time-series differs significantly from those of the source dataset. Additional training setup details are provided in Appendix D.

Key results. For each dataset, we compare our results to 5 representative entries reported in the literature for that dataset, based on dataset-specific metrics (M4, FRED, M3: sMAPE; TOURISM: MAPE; ELECTRICITY, TRAFFIC: ND). We additionally train the popular machine learning TS model DeepAR and evaluate it in the zero-shot regime. Our main results appear in Table 1, with more details provided in Appendix F. In the zero-shot forecasting regime (bottom three rows), N-BEATS consistently outperforms most statistical models tailored to these datasets as well as DeepAR trained on M4 and evaluated in zero-shot regime on other datasets. N-BEATS trained on FRED and applied in the zero-shot regime
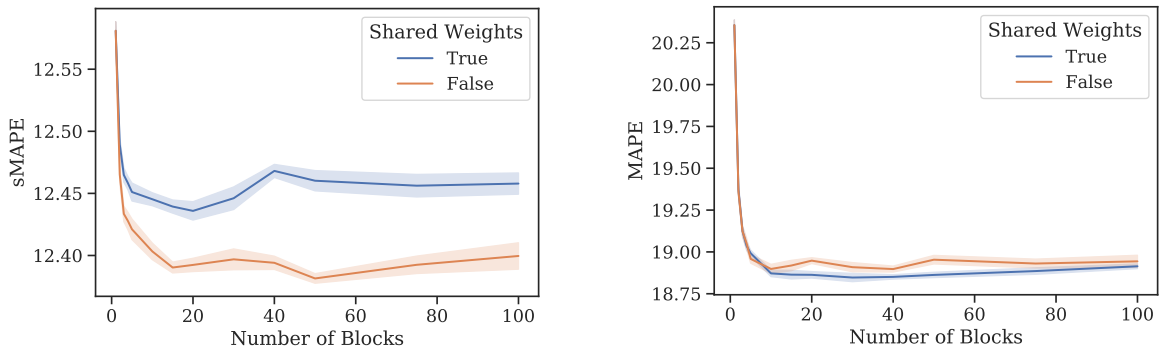
Figure 1: Zero-shot forecasting performance of N-BEATS trained on M4 and applied to M3 (left) and TOURISM (right) target datasets with respect to the number of blocks, L. The mean and one standard deviation interval (based on ensemble bootstrap) with (blue) and without (red) weight sharing across blocks are shown. The extended set of results for all datasets, using FRED as a source dataset and a few metrics are provided in Appendix G, further reinforcing our findings.

to M4 outperforms the best statistical model selected for its performance on M4 and is at par with the competition's second entry (boosted trees). On M3 and TOURISM the zero-shot forecasting performance of N-BEATS is better than that of the M3 winner, Theta (Assimakopoulos and Nikolopoulos 2000). On ELECTRICITY and TRAFFIC N-BEATS performs close to or better than other neural models trained on these datasets. The results suggest that a neural model is able to extract general knowledge about TS forecasting and then successfully adapt it to forecast on unseen TS. Our study presents the first successful application of a neural model to solve univariate zero-shot TS point forecasting across a large variety of datasets, and suggests that a pre-trained N-BEATS model can constitute a strong baseline for this task.

Meta-learning Effects. Analysis in Section 3 implies that N-BEATS internally generates a sequence of parameters that dynamically extend the expressive power of the architecture with each newly added block, even if the blocks are identical. To validate this hypothesis, we performed an experiment studying the zero-shot forecasting performance of N-BEATS with increasing number of blocks, with and without parameter sharing. The architecture was trained on M4 and the performance was measured on the target datasets M3 and TOURISM. The results are presented in Fig. 1. On the two datasets and for the shared-weights configuration, we consistently see performance improvement when the number of blocks increases up to about 30 blocks. In the same scenario, increasing the number of blocks beyond 30 leads to small, but consistent deterioration in performance. One can view these results as evidence supporting the meta-learning interpretation of N-BEATS, with a possible explanation of this phenomenon as overfitting in the meta-learning inner loop. It would not otherwise be obvious how to explain the generalization dynamics in Fig. 1. Additionally, the performance improvement due to meta-learning alone (shared weights, multiple blocks vs. a single block) is 12.60 to 12.44 (1.2%) and 20.40 to 18.82 (7.8%) for M3 and TOURISM, respectively (see Fig. 1). The performance improvement due to meta-learning and unique weights[1] (unique weights, multiple

blocks vs. a single block) is 12.60 to 12.40 (1.6%) and 20.40 to 18.91 (7.4%). Clearly, the majority of the gain is due to the meta-learning alone. The introduction of unique block weights sometimes results in marginal gain, but often leads to a loss (see more results in Appendix G).

In this section, we presented empirical evidence that neural networks are able to provide high-quality zero-shot forecasts on unseen TS. We further empirically supported the hypothesis that meta-learning adaptation mechanisms identified within N-BEATS in Section 3 are instrumental in achieving impressive zero-shot forecasting accuracy results.

5 Discussion and Conclusion

Zero-shot transfer learning. We propose a broad meta-learning framework and explain mechanisms facilitating zero-shot forecasting. Our results show that neural networks can extract generic knowledge about forecasting and apply it in zero-shot transfer. **Residual architectures** in general are covered by the analysis of Sec. 3, which might explain some of the success of residual architectures, although their deeper study should be subject to future work. Our theory suggests that residual connections generate, on-the-fly, compact task-specific parameter updates by producing a sequence of input shifts for identical blocks. Sec. 3.1 reinterprets our results showing that, as a first-order approximation residual connections produce an iterative update to the predictor final linear layer. **Memory efficiency and knowledge compression.** Our empirical results imply that N-BEATS is able to compress all the relevant knowledge about a given dataset in a single block, rather than in 10 or 30 blocks with individual weights. From a practical perspective, this could be used to obtain 10–30 times neural network weight compression and is relevant in applications where storing neural networks efficiently is important.

---

[1]Intuitively, the network with unique block weights includes the

network with identical weights as a special case. Therefore, it is free to combine the effect of meta-learning with the effect of unique block weights based on its training loss.

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.; Türkmen, A. C.; and Wang, Y. 2019. GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264* .

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

Athanasopoulos, G.; Hyndman, R. J.; Song, H.; and Wu, D. C. 2011. The tourism forecasting competition. *International Journal of Forecasting* 27(3): 822–844.

Bengio, S.; Bengio, Y.; Cloutier, J.; and Gecsei, J. 1992. On the optimization of a synaptic learning rule. In *Optimality in Artificial and Biological Neural Networks*.

Bengio, Y.; Bengio, S.; and Cloutier, J. 1991. Learning a Synaptic Learning Rule. In *Proceedings of the International Joint Conference on Neural Networks*, II–A969. Seattle, USA.

Chapados, N.; Joliveau, M.; L'Écuyer, P.; and Rousseau, L.-M. 2014. Retail store scheduling for profit. *European Journal of Operational Research* 239(3): 609 – 624.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. Last Accessed: 2021-03-01.

Engle, R. F. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50(4): 987–1007.

Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2018. Transfer learning for time series classification. *2018 IEEE International Conference on Big Data (Big Data)* .

Federal Reserve Bank of St. Louis. 2019. FRED Economic Data. Data retrieved from https://fred.stlouisfed.org/ Accessed: 2019-11-01.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, 1126–1135.

Fiorucci, J. A.; Pellegrini, T. R.; Louzada, F.; Petropoulos, F.; and Koehler, A. B. 2016. Models for optimising the theta method and their relationship to state space models. *International Journal of Forecasting* 32(4): 1151–1161.

Gao, J. 2014. Machine learning applications for data center optimization. Technical report, Google.

Gauss, C. F. 1809. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Hamburg: Frid. Perthes and I. H. Besser.

Harlow, H. F. 1949. The Formation of Learning Sets. *Psychological Review* 56(1): 51–65. doi:10.1037/h0062474.

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

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

Hooshmand, A.; and Sharma, R. 2019. Energy Predictive Models with Limited Data Using Transfer Learning. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy'19, 12–16.

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

Ibrahim, R.; Ye, H.; L'Ecuyer, P.; and Shen, H. 2016. Modeling and forecasting call center arrivals: A literature survey and a case study. *International Journal of Forecasting* 32(3): 865–874.

Kahn, K. B. 2003. How to Measure the Impact of a Forecast Error on an Enterprise? *The Journal of Business Forecasting Methods & Systems* 22(1).

Kerkkänen, A.; Korpela, J.; and Huiskonen, J. 2009. Demand forecasting errors in industrial context: Measurement and impacts. *International Journal of Production Economics* 118(1): 43–48.

Lee, Y.; and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2933–2942.

Leung, H. C. 1995. Neural networks in supply chain management. In *Proceedings for Operating Research and the Management Sciences*, 347–352.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-SGD: Learning to Learn Quickly for Few Shot Learning. *CoRR* abs/1707.09835.

M4 Team. 2018. M4 competitor's guide: prizes and rules. URL https://usermanual.wiki/Document/M4CompetitorsGuide.1491768831/view. Last Accessed: 2021-03-01.

Makridakis, S.; Andersen, A.; Carbone, R.; Fildes, R.; Hibon, M.; Lewandowski, R.; Newton, J.; Parzen, E.; and Winkler, R. 1982. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting* 1(2): 111–153.

Makridakis, S.; Chatfield, C.; Hibon, M.; Lawrence, M.; Mills, T.; Ord, K.; and Simmons, L. F. 1993. The M2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting* 9(1): 5–22.

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

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2018a. The M4-Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting* 34(4): 802–808.

Makridakis, S.; Spiliotis, E.; and Assimakopoulos, V. 2018b. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE* 13(3).

Montero-Manso, P.; Athanasopoulos, G.; Hyndman, R. J.; and Talagala, T. S. 2020. FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting* 36(1): 86–92.

Neale, A. A. 1985. Weather Forecasting: Magic, Art, Science and Hypnosis. *Weather and Climate* 5(1): 2–5.

Nguyen, H.-N.; Ni, Q.; and Rossetti, M. D. 2010. Exploring the cost of forecast error in inventory systems. In *Proceedings of the 2010 Industrial Engineering Research Conference*.

Oreshkin, B. N.; Carpov, D.; Chapados, N.; and Bengio, Y. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*.

Petersen, K. B.; and Pedersen, M. S. 2012. The Matrix Cookbook. Version 20121115.

Raghu, A.; Raghu, M.; Bengio, S.; and Vinyals, O. 2020. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In *International Conference on Learning Representations*.

Rangapuram, S. S.; Seeger, M.; Gasthaus, J.; Stella, L.; Wang, Y.; and Januschowski, T. 2018. Deep State Space Models for Time Series Forecasting. In *NeurIPS*.

Ravi, S.; and Larochelle, H. 2016. Optimization as a model for few-shot learning. In *ICLR*.

Ribeiro, M.; Grolinger, K.; ElYamany, H. F.; Higashino, W. A.; and Capretz, M. A. 2018. Transfer learning with seasonal and trend adjustment for cross-building energy forecasting. *Energy and Buildings* 165: 352–363.

Rodrigues Jr, F. A.; Jabloun, M.; Ortiz-Monasterio, J. I.; Crout, N. M. J.; Gurusamy, S.; and Green, S. 2019. Mexican Crop Observation, Management and Production Analysis Services System — COMPASS. In *Poster Proceedings of the 12th European Conference on Precision Agriculture*.

Rusu, A. A.; Rao, D.; Sygnowski, J.; Vinyals, O.; Pascanu, R.; Osindero, S.; and Hadsell, R. 2019. Meta-Learning with Latent Embedding Optimization. In *ICLR*.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2019. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* .

Schmidhuber, J. 1987. *Evolutionary principles in self-referential learning*. Master's thesis, Institut f. Informatik, Tech. Univ. Munich.

Sezer, O. B.; Gudelek, M. U.; and Ozbayoglu, A. M. 2019. Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019.

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.

Snell, J.; Swersky, K.; and Zemel, R. S. 2017. Prototypical Networks for Few-shot Learning. In *NIPS*, 4080–4090.

Spiliotis, E.; Assimakopoulos, V.; and Nikolopoulos, K. 2019. 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.

Tang, C.; and Salakhutdinov, R. R. 2019. Multiple Futures Prediction. In *NeurIPS 32*, 15398–15408.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. In *NIPS*, 3630–3638.

Walker, G. 1931. On Periodicity in Series of Related Terms. *Proc. R. Soc. Lond. A* 131: 518–532.

Wang, Y.; Smola, A.; Maddix, D. C.; Gasthaus, J.; Foster, D.; and Januschowski, T. 2019. Deep Factors for Forecasting. In *ICML*.

Winters, P. R. 1960. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science* 6(3): 324–342.

Yu, H.-F.; Rao, N.; and Dhillon, I. S. 2016. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *NIPS*.

Yule, G. U. 1927. On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers. *Phil. Trans. the R. Soc. Lond. A* 226: 267–298.