Deep Non-Parametric Time Series Forecaster

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper presents non-parametric baseline models for time series forecasting. Unlike classical forecasting models, the proposed approach does not assume any parametric form for the predictive distribution and instead generates predictions by sampling from the empirical distribution according to a *tunable* strategy. By virtue of this, the model is always able to produce reasonable forecasts (i.e., predictions within the observed data range) without fail unlike classical models that suffer from numerical stability on some data distributions. Moreover, we develop a global version of the proposed method that automatically learns the sampling strategy by exploiting the information across multiple related time series. The empirical evaluation shows that the proposed methods have reasonable and consistent performance across all datasets, proving them to be strong baselines to be considered in one's forecasting toolbox.

1 Introduction

Non-parametric models (or algorithms) are quite popular in machine learning for both supervised and unsupervised learning tasks especially in applied scenarios.[1] Examples of non-parametric models widely used in supervised learning [34] include classical k-nearest neighbours algorithm as well as more sophisticated tree-based methods like LightGBM and XGBoost [20, 7]. In contrast to parametric models which have a finite set of tunable parameters that do not grow with the sample size, non-parametric algorithms produce models that can become more and more complex with an increasing amount of data; e.g., decision surface learned by k-nearest neighbours and decision trees. One of the main advantages of non-parametric models is that they work on any dataset, produce reasonable baseline results and aid in developing more advanced models.

Despite the popularity of non-parametric methods in a general supervised setting, perhaps surprisingly, not much work is done in developing non-parametric methods in time series forecasting. Perretti et al. [31] show in a specific application setting that non-parametric methods can outperform parametric models in the presence of noise and advocate for more flexible non-parametric approaches. However, much of the work in time series forecasting has focussed on developing parametric models that typically assume Gaussianity of the data, e.g., ETS and ARIMA [15]. Some extensions of classical models have been proposed to handle intermittent data [8], count data [41], non-negative data [1] as well as more general non-Gaussian settings [39, 9]. However, most of these models cannot reliably work for all data distributions without running into numerical issues, which severely inhibits their usefulness in large-scale production settings. At the least, a robust, fail-safe model must be available to provide fall-back [6].

This paper attempts to bridge this gap in the literature by presenting non-parametric models for the forecasting problem that can reliably be used for any data distributions and generate reasonable probabilistic forecasts. For this, we first introduce a simple, *local* version of the proposed method

[1] See for example `https://www.kaggle.com/kaggle-survey-2020`.

called Non-Parametric Time Series Forecaster (NPTS, for short). The main idea here is to sample one of the time indices from the recent *context window* and use the value observed at that time index as the prediction for the next time step.[2] The sampling procedure is repeated to generate Monte Carlo samples of the predictive distribution, which is the standard way to represent forecast distribution [15]. We present different sampling strategies based on seasonality of the data. This way, the model can capture seasonality and its predictions are always within the observed data range and do not require any *overrides* in a production setting to filter out bad or outlier forecasts. Since, the strategy of sampling only from the past observations cannot capture trend, one can use the standard preprocessing techniques such as differencing (see e.g., [16]) to de-trend the data before applying NPTS. Since our standard benchmark data sets do not exhibit such a behaviour, we do not use this technique in our experiments.

We then develop a global extension of the method called Deep NPTS, where the sampling strategy is automatically learned from multiple related time series. For this we rely on a simple feed-forward neural network that takes in past data of all time series along with (optional) time series co-variates and outputs the sampling probabilities for the time indices. To train the model, we use a loss function based on the ranked probability score [11], which is a discrete version of the continuous ranked probability score (CRPS) [26]. Note that both RPS and CRPS are proper scoring rules for evaluating how likely the value observed is in fact generated from the given distribution [11, 14]. Once the model is trained, the sampling probabilities and then the predictions can be obtained by doing a forward pass on the feed-forward neural network, which does not result in any numerical problems. In spite of being a deep-learning based model, Deep NPTS produces output that is explainable (Figure 1). Moreover, it generates calibrated forecasts (Figure 2) and for non-Gaussian settings like integer data or rate/interval data, NPTS in general and DeepNPTS in particular give much better results than the standard baselines, suggesting that our methods are good baselines to consider.

The article is structured as follows. We introduce the local model, NPTS, in Section 2 and then extend it to a global version in Section 3. After discussing related work in Section 4, we provide both qualitative and extensive quantitative experiments in Section 5. We conclude in Section 6.

2 Non-Parametric Time Series Forecaster

Here we introduce the non-parametric forecasting method for a single univariate time series. This method is *local* in the sense that it will be applied to each time series independently, similar to the classical models like ETS and ARIMA. In Section 3 we discuss a global version, which is more relevant to modern time series panels.

Let $z_{0:T-1} = (z_0, z_1, \ldots, z_{T-1})$ be a given univariate time series. The time series z is univariate if each z_i is a one-dimensional value. Note that we do not need to further specify the domain of the time series (e.g., whether $z \in \mathbb{R}^T$ or $z \in \mathbb{Z}^T$) as this is not important unlike for other methods. To generate prediction for the next time step T, NPTS randomly samples a time index $t \in \{0, \ldots, T-1\}$ from the past observed time range and use the value observed at that time index as the prediction. That is,

$$\hat{z}_T = z_t, \quad t \sim p_T(\cdot), \quad t \in \{0, \ldots, T-1\},$$

where $p_T(\cdot)$ is a categorical distribution with T states. To generate distribution forecast, we sample time indices from $p_T(\cdot)$ K times and store the corresponding observations as Monte Carlo samples of the predictive distribution.

Note that this is quite different from naive forecasting methods that choose a fixed time index to generate prediction, e.g., $T-1$ or $T-\tau$, where τ represents seasonal period. The sampling of time indices instead of choosing a fixed index from the past immediately brings up two advantages:

- it makes the forecaster probabilistic and consequently allows one to generate prediction intervals,
- it gives the flexibility by leaving open the choice of sampling distribution p_T; one can define p_T based on prior knowledge (e.g., seasonality) or in more generally learn it from the data.

[2]This is similar in spirit to k-nearest neighbours algorithm but unlike that method, our approach naturally generates a probabilistic output.

We now discuss some choices of sampling distribution $p_T(\cdot)$ that give rise to various specializations of NPTS. In Section 3 we discuss how to learn the sampling distribution.

One natural idea for the sampling distribution is to weigh each time index of the past according to its "distance" from the time step where the forecast is needed. An obvious choice is to use the exponentially decaying weights as the recent past is more indicative of what is going to happen in the next step. This results in what we call NPTS (without any further qualifications):

$$p_T(t) \propto \exp(-\alpha\,|T-t|),$$

where α is a hyper-parameter that will be tuned based on the data. We also refer to this variant as NPTS with exponential kernel.

Seasonal NPTS: One can generalize the notion of distance from simple time indices to time-based features $f(t) \in \mathbb{R}^D$, e.g., hour-of-day, day-of-week. This results in what we call seasonal NPTS

$$p_T(t) \propto \exp(-\alpha\,|f(T)-f(t)|).$$

The feature map f can in principle be learned as well from the data. In this case, one keeps the exponential kernel intact but only learns the feature map. The method presented in the next section directly learns p_T.

NPTS with uniform kernel (Climatological Forecaster): The special case, where one uses uniform weights for all the time steps in the context window, i.e., $p_T(t) = 1/T$, leads to Climatological forecaster [14]. One can similarly define a seasonal variant by placing uniform weights only on past seasons indicated by the feature map. Again, this is different from the seasonal naive method [16], which uses the observation from the latest season alone as the prediction whereas the former uniformly samples from several seasons to generate prediction.

Extension to Multi-Horizon Forecast: Note that so far we only talked about generating one step ahead forecast. To generate forecasts for multiple time steps one simply absorbs the predictions for last time steps into the observed targets and then generates consequent predictions using the last T targets. More precisely, let $\{\hat{z}_{T,k}\}_{k=1}^{K}$ be the prediction samples obtained for the time step T. Then prediction samples for time steps $T+t,\ t>0$ are generated auto-regressively using the values $(z_t,\ldots,z_{T-1},\hat{z}_{T,k},\ldots,\hat{z}_{T+t-1,k}), k=1,2,\ldots,K$.

3 Deep Non-Parametric Time Series Forecaster

The main idea of Deep NPTS model is to learn the sampling distribution from the data itself and continue to sample only from the observed values. Let N be a set of univariate time series $\{z_{0:T_i-1}^{(i)}\}_{i=1}^{N}$, where $z_{1:T_i-1}^{(i)} = (z_1^{(i)}, z_2^{(i)}, \ldots, z_{T_i-1}^{(i)})$ and $z_t^{(i)}$ is a scalar quantity denoting the value of the i-th time series at time t (or the time series of *item* i).[3] Further, let $\{\mathbf{x}_{0:T_i}^{(i)}\}_{i=1}^{N}$ be a set of associated, time-varying covariate vectors with $\mathbf{x}_t^{(i)} \in \mathbb{R}^D$. We can assume time-varying covariate vectors to be the only type of co-variates without loss of generality, as time-independent and item-specific features can be incorporated into $\mathbf{x}^{(i)}$ by repeating the feature value over all time points. Our goal is to learn the sampling distribution $p_{T_i}^{(i)}(t), t=0,\ldots,T_i-1$ w.r.t. to the forecast start time T_i for each time series. The prediction for time step T_i for the i^{th} time series is then obtained by sampling $p_{T_i}^{(i)}$.

3.1 Model

Here we propose to use a feed-forward neural network to learn the sampling probabilities. As inputs to this global model, we define a fixed length *context window* of size T spanning the last T observations $z_{T_i-1:T_i-T}^{(i)}$. In the following, without loss of generality, we refer to this context window as the interval $[0, T-1]$ and the prediction time step as T; however, note that the actual time indices corresponding to this context window would be different for different time series.

[3] We consider time series where the the time points are equally spaced, but the units or frequencies are arbitrary (e.g. hours, days, months). Further, the time series do not have to be aligned, i.e., the starting point $t=1$ can refer to a different absolute time point for different time series i.

For each time series i, given the observations from the context window, the network outputs the sampling probabilities to be used for prediction for time step T. More precisely,

$$p_T^{(i)}(t) = \Psi(\mathbf{x}_{0:T}^{(i)}, z_{0:T-1}^{(i)}; \phi),\ t = 0, 1, \dots, T-1. \tag{1}$$

Here Ψ the neural network and ϕ is the set of neural network weights that are shared among all the time series. Note that for each time series i, the network outputs potentially different sampling probabilities if time series features differ among the time series. However, these different sampling probabilities are parametrized by a single set of common parameters ϕ, facilitating information sharing and global learning from multiple time series.

We assume from here onwards that the outputs of the network are normalized so that $p_T^{(i)}(t)$ represent probabilities. This can be achieved by using the softmax activation function for the final layer or using the standard normalization (i.e., dividing each output by the sum of the outputs). It turned out that either of the two normalizations do not consistently give the best possible results for all the datasets. Hence we treat this as a hyper-parameter in our experiments.

3.2 Training

We now describe the training procedure for our model defined in Eq. 1, in particular by defining an appropriate loss function. For ease of exposition and without loss of generality, we drop the index i in this section. Our prediction $\hat{z}_T$ for a given univariate time series $(z_0, z_1, \dots, z_{T-1})$ at time step T is generated by sampling from p_T. So our forecast distribution for time step T can be seen as sampling from the discrete random variable $\hat{Z}_T$ with the probability mass function given by

$$f_{\hat{Z}_T}(z_t) = \sum_{t': z_{t'} = z_t} p_T(t'), \quad t = 0, \dots, T-1. \tag{2}$$

That is, the prediction is always one of $z_t, t = 0, \dots, T-1$ and the probability of predicting z_t is the sum of the sampling probabilities of those time indices where the value z_t is observed. Similarly, the cumulative distribution function for any value z is given by

$$F_{\hat{Z}_T}(z) = \sum_{t: z_t \le z} p_T(t). \tag{3}$$

Loss: Ranked Probability Score. Given that our forecasts are generated by sampling from the discrete random variable $\hat{Z}_T$, we propose to use the ranked probability score between our probabilistic prediction (specified by $F_{\hat{Z}_T}$) and the actual observation z_T,

$$\text{RPS}(F_{\hat{Z}_T}, z_T) = \sum_{z_t \in \{z_0, \dots, z_{T-1}\}} \Lambda_{\alpha_t}(z_t, z_T), \tag{4}$$

where $\alpha_t = F_{\hat{Z}_T}(z_t)$ is the quantile level of z_t and $\Lambda_\alpha(q, z)$ is the quantile loss given by

$$\Lambda_\alpha(q, z) = (\alpha - \mathcal{I}_{[z<q]})(z - q).$$

Note that the summation in the loss Eq. 4 runs only on the distinct values of the past observations given by the *set* $\{z_0, \dots, z_{T-1}\}$. The total loss of the network with parameters ϕ over all the training examples $\{z_{0:T}^{(i)}\}$ can then be defined as

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} \text{RPS}(F_{\hat{Z}_T^{(i)}}, z_T^{(i)}) \tag{5}$$

Data Augmentation: Note that the loss defined in Eq. 5 uses only a single time step T to evaluate the prediction, for each training example. Since the same model would be used for multi-step ahead prediction, we generate multiple training instances from each time series by selecting context windows with different starting time points, similar to [38]. For example, assume that the training data is available from February 01 to February 21 of a daily time series and we are required to predict for 7 days. We can define the context window to be of size, say, 14 and generate 7 training examples with the following sliding context windows: February $1 + k : 14 + k, k = 0, 1, \dots, 6$. For each of these seven context windows (which are passed to the network as inputs), the training loss is computed using the observations at February $15 + k, k = 0, 1, \dots, 6$ respectively. We do the same for all the time series in the dataset.

3.3 Prediction

Once the model is trained, it can generate forecast distribution for a single time step. The multi-step ahead forecast can then be generated as described in the previous section using the same trained model. Note that to generate multi-horizon forecasts one needs to have access to time series feature values for the future time steps, a typical assumption of global models in time series forecasting [38, 33].

4 Related Work

A number of new time series forecasting methods have been proposed over the last years, in particular global deep learning methods (e.g., [29, 38, 23, 30, 36]). Benidis et al. [4] provides a recent overview. The usage of global models [19] in forecasting has well-known predecessors (e.g., [13] and [44] for a modern incarnation), however local models have traditionally dominated forecasting which have advantages given their parsimonious parametrization, interpretability and stemming from the fact that many time series forecasting problems consists of few time series. The surge of global models in the literature can be explained both by their theoretical superiority [28] as well as empirical success in independent competitions [25, 5]. We believe that both, local and global methods will continue to have their place in forecasting and its practical application. For example, the need for fail-safe fall-back models in real-world production use-cases has been recognized [6]. Therefore, the methods presented here have both local and global versions.

While many methods of the afore-mentioned recent global deep learning methods only consider providing point forecasts, some do provide probabilistic forecasts [45, 38, 33] motivated by downstream decision making problems often requiring the minimization of expected cost (e.g., [40]). The approaches to produce probabilistic forecast range from making standard parametric assumptions on the pdf (e.g., [38, 32]), to more flexible parametrizations via copulas [37] or normalizing flows [35, 9], sometimes using extensions to energy-based models [36], from quantile regression [45] to parametrization of the quantile function [12]. All these approaches share an inherent risk induced by potential numerical instability. For example, even estimating a standard likelihood of a linear dynamical system via a Kalman Filter (see [33] for a recent forecasting example) easily results in numerical complications unless care is taken. In contrast, the proposed methods here are almost completely fail-safe in the sense that numerical issues will not result in catastrophically wrong forecasts. While the added robustness comes at the price of some accuracy loss, the overall accuracy is nevertheless competitive and an additional benefit is the reduced amount of hyperparameter tuning necessary, even for Deep NPTS.

The general idea of constructing predictive distributions from the empirical distributions of (subsets of) observations has been explored in the context of probabilistic regression, e.g. in the form of Quantile Regression Forests [27] or more generally in the form of conformal prediction [43].

5 Experiments

We present empirical evaluation of the proposed method on the following datasets, which are publicly available in `GluonTS` time series library [2].

- Electricity: hourly time series of the electricity consumption of 370 customers [10]
- Exchange rate: daily exchange rate between 8 currencies as used in [21]
- Solar: hourly photo-voltaic production of 137 stations in Alabama State used in [21]
- Taxi: spatio-temporal traffic time series of New York taxi rides [42] taken at 1214 locations every 30 minutes in the months of January 2015 (training set) and January 2016 (test set)
- Traffic: hourly occupancy rate of 963 San Francisco car lanes [10]
- Wikipedia: daily page views of 9535 Wikipedia pages used in [12]
- M4: datasets, of varying frequencies from hourly to yearly, used in the M4 competition [25]

Table 1 summarizes the key features of these datasets. For M4 datasets there is a singe multi-horizon prediction window where the forecasts are evaluated. For other datasets, the predictions are evaluated in a rolling-window fashion. The length of the prediction window as well as the number

dataset	No. Test Points Pred. Len. × No. Windows	Domain	Freq.	Size	(Median) TS Len.
`Exchange Rate`	150 (30 × 5)	$\mathbb{R}^+$	daily	8	6071
`Solar Energy`	168 (24 × 7)	$\mathbb{R}^+$	hourly	137	7009
`Electricity`	168 (24 × 7)	$\mathbb{R}^+$	hourly	370	5790
`Traffic`	168 (24 × 7)	$[0, 1]$	hourly	963	10413
`Taxi`	1344 (24 × 56)	$\mathbb{N}$	30-min	1214	1488
`Wiki`	150 (30 × 5)	$\mathbb{N}$	daily	2000	792
`M4 hourly`	48 × 1	$\mathbb{R}^+$	hourly	414	960
`M4 daily`	14 × 1	$\mathbb{R}^+$	daily	4227	2940
`M4 weekly`	13 × 1	$\mathbb{R}^+$	weekly	359	934
`M4 monthly`	18 × 1	$\mathbb{R}^+$	monthly	48000	202
`M4 quarterly`	8 × 1	$\mathbb{R}^+$	quarterly	24000	88
`M4 yearly`	6 × 1	$\mathbb{R}^+$	yearly	23000	29

Table 1: Summary of the datasets used in the evaluations: Number of time ssteps evaluated, data domain, frequency of observations, number of time series in the dataset, and median length of time series.

of such windows are given in Table 1. Note that for each time series in a dataset, forecasts are evaluated at a total of τ time points, where τ is length of the prediction window times the number of windows. For τ, the total evaluation length, let $T + \tau$ be the length of the time series available for a dataset. Then each method initially receives time series for the first T time steps which are used to tune hyperparameters in a back-test fashion, e.g., training on the first $T - \tau$ steps and validating on the last τ time steps. Once the best hyperparameters are found, each model is once again trained on T time steps and is evaluated on the time steps from $T + 1$ to $T + \tau$.

Evaluation Criteria: For evaluating the forecast distribution, we use the mean of quantile losses evaluated at different quantiles implemented in `GluonTS`, with the quantile levels ranging from 0.5 to 0.95 in steps of 0.05. Note that this is an approximate version of the continuously ranked probability score (CRPS), a proper metric for evaluating predictive distributions. Additionally, to evaluate mean forecasts, we use the (normalized) root-mean squared error (RMSE), which is an evaluation metric for point forecasts.

Parameter	Range
`dropout`	$\{0, 0.1\}$
`static feat`	`True | Flase`
`normalization`	`softmax, normal`
`input scaling`	`None, standardization`
`loss scaling`	`None, min/max scaling`
`epochs`	$\{200, 300\}$

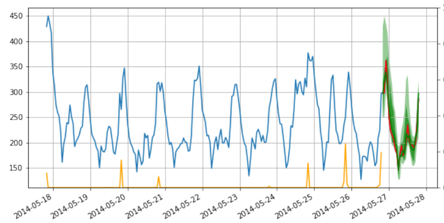
Table 2: Hyperparameter grid for `DeepNPTS`.

Parameter	Range
`dropout`	$\{0, 0.1\}$
`static feat`	`True | Flase`
`num of RNN cells`	$\{40, 80\}$
`context length`	$\{1, 2\} \times$ `pred. length`
`epochs`	$\{200, 300\}$

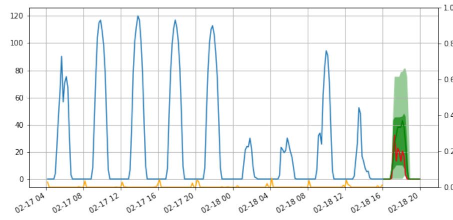
Table 3: Hyperparameter grid for `DeepAR`.

Methods Compared: We compare against the standard baselines in forecasting literature including `Seasonal-Naive`, `ETS`, `ARIMA` [16] as well as `DeepAR` [38], a deep-learning based forecasting model that has shown to be one of the best performing models empirically [2][4]. We also include for comparison all the variants of the proposed NPTS method. In particular, we have the following four combinations: (i) `NPTS` with or without seasonality (ii) `NPTS` using uniform weights or exponentially decaying weights. For the variants of `NPTS` that use exponential kernel (`NPTS`, `Seas.NPTS`) we tune the (inverse of) width parameter α on the validation set. In particular, we use the grid: $\alpha \in \{1.0, 0.75, 0.5, 0.25, 0.1\}$. For the other two variants using the uniform kernel (`NPTS(uni.)`, `Seas.NPTS(uni.)`), there are no hyperparameters to be tuned. For the `DeepNPTS` model the hyperparameter grid is given in Table 2. The number of layers of the MLP is fixed at 2 and the number of hidden nodes (equal to the size of the context window, see Section 3.1) is chosen as a constant multiple of the prediction length. This multiple varies for each dataset (depending on the length of the

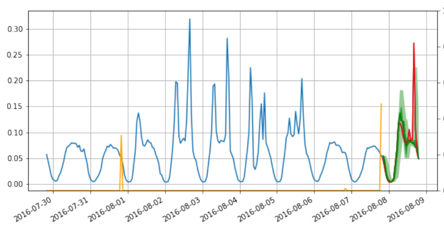
[4]Results for more baselines are given in the supplementary material.



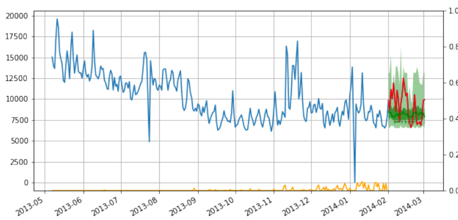
(a) `Electricity`



(b) `Solar Energy`



(c) `Traffic`



(d) `Wiki`

Figure 1: Example of forecasts obtained with DeepNPTS on some of the datasets used for experiments, together with the probabilities that the model outputs. The dark green line is the median forecast, surrounded by the 50% (green) and 90% (light green) prediction intervals. The red line is the actual target in the forecast time window. The orange line indicates the probability assigned by the model to the corresponding time point (as measured on the dual axis on the right of each plot), in making the *first* time step prediction: this highlights the seasonal patterns that the model finds in the data.

time series available) and is fixed at a large value, without tuning, so that multiple training instances (equal to the prediction length) can be generated for each time series in the dataset as discussed in Section 3.1. We give the exact lengths of the context windows used for each dataset in the supplementary material. Similarly Table 3 shows the hyperparameter grid for the `DeepAR` method. The other hyper-parameters of `DeepAR` are fixed at the default values [2]. Both `DeepAR` and `DeepNPTS` receive time features that are automatically determined based on the frequency of the given dataset as implemented in `GluonTS` [2]. For `ETS` and `ARIMA`, we use the auto-tuning option provided by the R `forecast` package [18]. We use the mean quantile loss as the criteria for tuning the models, since all models compared here produce distribution forecasts.

Qualitative Analysis: Before presenting quantitative results, we first analyse the performance `DeepNPTS` model qualitatively by visualizing its forecasts as well as the probabilities it learned, by considering specific time series in detail (instead of overall aggregate accuracy metrics). Figure 1 shows forecasts of `DeepNPTS` for one example time series taken from each of the four of the datasets used in the experiments. Each plot shows the true target as well as the 50% and 90% prediction intervals of the forecast distribution for the prediction window. Additionally, we show the output of the model (i.e., sampling probabilities) with an orange line (note the dual axes on the right side of each plot). Note that this is the output of the model in making the prediction for the *first* time step of the prediction window. In the case of `Traffic`, one can notice that the last observation and the same hour on the previous week received the highest probabilities, indicating that the model correctly captures the hour-of-week seasonality pattern. For `Electricity`, the same hour over multiple consecutive days was assigned high probability, indicating a hour-of-day seasonality. For non-seasonal time series like `Wiki` the most recent time points are assigned higher probabilities.

One of the main benefits of `DeepNPTS` is then the *explainability* of its output: although it is a deep-learning based model, one can easily explain how the model generated forecasts by looking at which time points got higher probabilities, for any given time step, and also verify if the model assigned probabilities as intended.

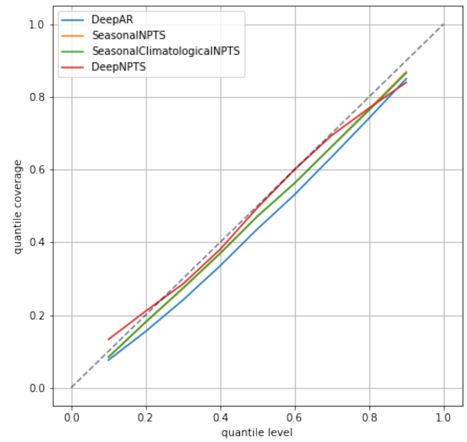


Figure 2: Calibration plot for the `Traffic` dataset: this shows what fraction of the actual data lies below the predicted quantiles, and is a crucial quality metric for probabilistic forecasts (the ideal profile lies on the diagonal). On this example, the predictions from `DeepNPTS` appear to be better calibrated than the ones from `Seas.NPTS` and `Seas.NPTS(uni.)`, which are in turn better calibrated than the ones from `DeepAR`.

dataset estimator	`Electricity`	`Exchange Rate`	`Solar Energy`	`Taxi`	`Traffic`	`Wiki`
`Seas.Naive`	0.070+/-0.000	0.011+/-0.000	0.605+/-0.000	0.507+/-0.000	0.251+/-0.000	0.404+/-0.000
`AutoETS`	0.118+/-0.001	**0.007+/-0.000**	1.717+/-0.005	0.572+/-0.000	0.359+/-0.000	0.664+/-0.001
`AutoARIMA`	-	**0.008+/-0.000**	1.161+/-0.001	0.473+/-0.000	-	0.477+/-0.000
`DeepAR`	**0.056+/-0.001**	0.009+/-0.001	**0.426+/-0.004**	**0.289+/-0.001**	**0.117+/-0.001**	**0.226+/-0.002**
`NPTS(uni.)`	0.230+/-0.001	0.026+/-0.000	0.805+/-0.001	0.473+/-0.000	0.403+/-0.000	0.291+/-0.000
`Seas.NPTS(uni.)`	0.061+/-0.000	0.026+/-0.000	0.791+/-0.000	0.474+/-0.000	0.174+/-0.000	0.285+/-0.000
`NPTS`	0.230+/-0.001	0.021+/-0.000	0.802+/-0.001	0.473+/-0.000	0.403+/-0.000	0.277+/-0.000
`Seas.NPTS`	0.060+/-0.000	0.020+/-0.000	0.790+/-0.000	0.474+/-0.000	0.173+/-0.000	0.269+/-0.000
`DeepNPTS`	**0.059+/-0.004**	0.009+/-0.000	**0.447+/-0.011**	**0.413+/-0.015**	**0.163+/-0.036**	**0.232+/-0.000**

Table 4: Mean quantile losses averaged over 5 runs (lower is better). Best two methods are highlighted. The bottom half shows different variants of the NPTS method. `AutoARIMA` exceeded the time limit of 24 hours for `Electricity` and `Traffic` data sets, so we do not report their numbers.

Next, we analyse how calibrated are the forecasts of the `DeepNPTS` model. Figure 2 shows calibration plot for the `Traffic` dataset. Calibration plot shows what fraction of the actual data lies below the predicted quantiles, with the ideal profile being the diagonal line. It is a crucial quality metric for probabilistic forecasts. As expected, forecasts from both the `NPTS` and `DeepNPTS` models are highly calibrated in absolute terms and in relative terms, more so than `DeepAR` with a student-t output distribution.

dataset	`M4 daily`	`M4 hourly`	`M4 monthly`	`M4 quarterly`	`M4 weekly`	`M4 yearly`
`Seas.Naive`	0.028+/-0.000	0.048+/-0.000	0.146+/-0.000	0.119+/-0.000	0.063+/-0.000	0.162+/-0.000
`AutoETS`	**0.022+/-0.000**	**0.042+/-0.001**	**0.096+/-0.000**	**0.076+/-0.000**	**0.049+/-0.000**	**0.122+/-0.000**
`AutoARIMA`	0.024+/-0.000	**0.038+/-0.001**	**0.094+/-0.000**	**0.077+/-0.000**	**0.048+/-0.000**	**0.120+/-0.000**
`DeepAR`	**0.022+/-0.000**	0.050+/-0.003	0.112+/-0.002	0.086+/-0.004	0.051+/-0.004	0.130+/-0.007
`NPTS(uni.)`	0.161+/-0.000	0.115+/-0.001	0.257+/-0.000	0.289+/-0.000	0.319+/-0.001	0.389+/-0.000
`Seas.NPTS(uni.)`	0.161+/-0.000	0.053+/-0.000	0.258+/-0.000	0.288+/-0.000	0.329+/-0.001	0.389+/-0.000
`NPTS`	0.139+/-0.000	0.112+/-0.001	0.224+/-0.000	0.246+/-0.000	0.272+/-0.000	0.343+/-0.000
`Seas.NPTS`	0.139+/-0.000	0.046+/-0.000	0.225+/-0.000	0.245+/-0.000	0.285+/-0.000	0.343+/-0.000
`DeepNPTS`	0.027+/-0.000	0.065+/-0.019	0.149+/-0.012	0.097+/-0.002	0.057+/-0.001	0.168+/-0.008

Table 5: Mean quantile losses averaged over 5 runs (lower is better). Best two methods are highlighted. The bottom half shows different variants of the NPTS method.

Quantitative Results: Tables 4 and 5 summarize the quantitative results for the two groups of datasets considered via mean quantile losses, averaged over 5 runs. Top two best performing methods are highlighted with **boldface**. In both tables, the top half shows the baselines considered and the bottom half shows different variants of the proposed NPTS method; more baseline results are in the supplement. First note that `DeepNPTS` model always (except for one case) achieves much better results than any other variant of `NPTS` showing that learning the sampling strategy clearly helps. Moreover, `DeepNPTS` comes as one of the top two methods in 5 out of 12 datasets considered and in the remaining its results are close to the best performing method. Importantly for the *difficult* datasets like `Solar Energy` (non-negative data with several zeros), `Traffic` (data lies in $(0, 1)$) and `Wiki` (integer data), the gap between the performance of `DeepNPTS` and the standard baselines `AutoETS` and `AutoARIMA` is very high. Even some of the other variants of `NPTS` (with fixed sampling strategy) performed better than `AutoETS` and `AutoARIMA` in these datasets without any training (other than the tuning of α). All the variants of `NPTS` run much faster than the standard baselines `AutoETS`, `AutoARIMA`, which fit a different model to each time series in the dataset. These observations, in addition to explainability and being able to generate calibrated forecasts, further support our claim that the prosed methods in general and `DeepNPTS` in particular are good baselines to consider for any data distributions. In the supplement, we include additional results evaluating the mean (point) forecasts using the RMSE metric; the results trend is exactly the same.

6 Conclusion

In this paper we presented a novel probabilistic forecasting method, in both an extremely simple yet effective local version, and an adaptive, deep-learning-based global version. In both variants, the proposed methods can serve as robust, fail-safe forecasting methods that are able to provide accurate probabilistic forecasts. We achieve robustness by constructing the predictive distributions by reweighting the empirical distribution of the past observations, and achieve accuracy by taking advantage of learning the context-dependent weighting globally, across time series. We show in empirical evaluations that the methods, NPTS and Deep NPTS, perform roughly on-par with recent, state-of-the-art methods both quantitatively and qualitatively.

Limitations: One key limitation of the proposed approach is that it—like all methods that rely directly on the empirical distributions of the observations to construct predictive distributions—without additional processing cannot model non-stationary time series (e.g. containing trend). While techniques for achieving stationarity (like differencing and de-trending) are readily available, care must be taken when they are combined with our proposed approach to retain robustness. Addressing this limitation in a principled way is an interesting avenue for future work. Further, the predictive distributions do not have a compact parametric form (or even a density), precluding certain applications (though both could be obtained post-hoc e.g. through a parametric fit or smoothing via kernel density estimation). Finally, a limitation of this work stems from its intention as a robust baseline method: we do not expect our method to enable a truly state-of-the-art accuracy without introducing additional elements that would violate one of the design principles such as robustness.

7 Broader Impact

The present article stems from the authors' work on time series forecasting in industrial settings. The proposed methods are applicable to forecasting and allow to re-use multiple time series panels. Business applications include supply chain optimization, sales prediction and energy forecasting. More accurate forecasts allow for better decision making which we hope will lead to benefits such as waste reduction, reduction of emission through improve transportation costs and optimisation of energy consumption.

References

[1] M. Akram, J. Hyndman, and K. Ord. Exponential smoothing and non-negative data. 51(4): 415–432, 2009.

[2] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, and J. Schulz. GluonTS: Probabilistic Time Series Models in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

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

[4] K. Benidis, S. S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, and T. Januschowski. Neural forecasting: Introduction and literature overview, 2020.

[5] C. S. Bojer and J. P. Meldgaard. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2):587–603, 2021. ISSN 0169-2070.

[6] J.-H. Böse, V. Flunkert, J. Gasthaus, T. Januschowski, D. Lange, D. Salinas, S. Schelter, M. Seeger, and Y. Wang. Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10(12):1694–1705, 2017.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794. ACM, 2016.

[8] J. Croston. Forecasting and stock control for intermittent demands. *Journal of Operational Research Quarterly*, 23(3):289–303, 1972.

[9] E. de Bézenac, S. S. Rangapuram, K. Benidis, M. Bohlke-Schneider, R. Kurle, L. Stella, H. Hasson, P. Gallinari, and T. Januschowski. Normalizing kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 2995–3007, 2020.

[10] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[11] E. S. Epstein. A scoring system for probability forecasts of ranked categories. *J. Appl. Meteor.*, 8:985–987, 1969.

[12] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski. Probabilistic forecasting with spline quantile function RNNs. *AISTATS*, 2019.

[13] J. Geweke. The dynamic factor analysis of economic time series. *Latent variables in socio-economic models*, 1977.

[14] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[15] R. Hyndman, A. Koehler, K. Ord, and R. Snyder. *Forecasting with exponential smoothing. The state space approach*. 2008. doi: 10.1007/978-3-540-71918-2.

[16] R. J. Hyndman and G. Athanasopoulos. Forecasting: Principles and practice. *www.otexts.org/fpp*, 987507109, 2017.

[17] R. J. Hyndman and B. Billah. Unmasking the Theta method. *International Journal of Forecasting*, 19(2):287–290, 2003.

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

[19] T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 2019.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[21] G. Lai, W. Chang, Y. Yang, and H. Liu. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017.

[22] E. Liberty, Z. Karnin, B. Xiang, L. Rouesnel, B. Coskun, R. Nallapati, J. Delgado, A. Sadoughi, Y. Astashonok, P. Das, C. Balioglu, S. Chakravarty, M. Jha, P. Gautier, D. Arpin, T. Januschowski, V. Flunkert, Y. Wang, J. Gasthaus, L. Stella, S. Rangapuram, D. Salinas, S. Schelter, and A. Smola. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 731–737, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356.

[23] B. Lim, S. O. Arik, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2019.

[24] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[25] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[26] J. E. Matheson and R. L. Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

[27] N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun): 983–999, 2006.

[28] P. Montero-Manso and R. J. Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 2021.

[29] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[30] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *arXiv preprint arXiv:2002.02887*, 2020.

[31] C. Perretti, G. Sugihara, and S. Munch. Forecasting and stock control for intermittent demands. *Ecology*, 94(4):794–800, 2013.

[32] S. Rabanser, T. Januschowski, V. Flunkert, D. Salinas, and J. Gasthaus. The effectiveness of discretization in forecasting: An empirical study on neural time series models. *arXiv preprint arXiv:2005.10111*, 2020.

[33] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[34] S. Raschka. Machine learning FAQ. What is the difference between a parametric learning algorithm and a nonparametric learning algorithm? `https://sebastianraschka.com/faq/docs/parametric_vs_nonparametric.html`.

[35] K. Rasul, A.-S. Sheikh, I. Schuster, U. Bergmann, and R. Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*, 2020.

[36] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *arXiv preprint arXiv:2101.12072*, 2021.

[37] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems 32*, 2019.

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

[39] M. W. Seeger, D. Salinas, and V. Flunkert. Bayesian intermittent demand forecasting for large inventories. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/03255088ed63354a54e0e5ed957e9008-Paper.pdf`.

[40] D. Simchi-Levi and E. Simchi-Levi. We need a stress test for critical supply chains. *Harvard Business Review*, 2020.

[41] R. D. Snyder, J. K. Ord, and A. Beaumont. Forecasting the intermittent demand for slow-moving inventories: A modelling approach. *International Journal of Forecasting*, 28(2):485–496, 2012.

[42] N. Taxi and L. Commission. TLC trip record data. `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, 2015.

[43] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387001522.

[44] Y. Wang, A. Smola, D. Maddix, J. Gasthaus, D. Foster, and T. Januschowski. Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617, 2019.

[45] R. Wen, K. Torkkola, and B. Narayanaswamy. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will open source the code after acceptance, we included instruction to reproduce the results and we used datasets available online.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? See supplementary material. [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We will only open-source the code later (not before, in order not to break anonymity of the submission).
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] These are numerical time series in an aggregated fashion so this does not apply.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Experiment Details & Results

Here we provide further details of our experimental set up, report additional results evaluating our method against two more baselines and using one more evaluation metric that measures error for point forecasts.

All the deep learning based methods are run using Amazon SageMaker [22] on a machine with 3.4GHz processor and 32GB RAM. The remaining methods (R-based) are run on Amazon cloud instance of same configuration, 3.4GHz processor and 32GB RAM.

Detailed information about datasets and how to download them is already provided in the main text; see Section 5. Moreover, details of hyper-parameters and their tuning is also provided in the experiments section. We used the code available in `GluonTS` forecasting library to run all the baseline methods compared. Note that our method `DeepNPTS` generates multiple training instances for each time series as explained in Section 3 (see "Data Augmentation" paragraph). We use `GluonTS` for this purpose and fix the length of the context window large enough such that the required number of training instances can be generated. So we fix the context length depending on the history of the time series available, more importantly, without tuning. Table 6 shows the context lengths (as factor of prediction lengths) used for different datasets.

dataset	context Length
M4 daily	$14\times$ pred. length
M4 hourly	$10\times$ pred. length
M4 monthly	$2\times$ pred. length
M4 quarterly	$2\times$ pred. length
M4 yearly	$1\times$ pred. length
Wiki	$10\times$ pred. length
Other datasets	$28\times$ pred. length

Table 6: Context lengths used for `DeepNPTS`.

A.1 Quantitative Results

In addition to the standard baselines used in the main version of the paper, here we include comparisons against `TBATS` [24] and `Theta` [3] methods. `TBATS` incorporates Box-Cox transformation and Fourier representations in the state space framework to handle complex seasonal time series thereby addressing the limitations of `ETS` and `ARIMA`. `Theta` forecasting method is an additional baseline with good empirical performance [3] and its forecasts are equivalent to simple exponential smoothing model with drift [17]. The mean quantile losses for all the methods is shown in Tables 7 and 8.

dataset estimator	Electricity	Exchange Rate	Solar Energy	Taxi	Traffic	Wiki
Seas.Naive	0.070+/-0.000	0.011+/-0.000	0.605+/-0.000	0.507+/-0.000	0.251+/-0.000	0.404+/-0.000
AutoETS	0.118+/-0.001	**0.007+/-0.000**	1.717+/-0.005	0.572+/-0.000	0.359+/-0.000	0.664+/-0.001
AutoARIMA	-	0.008+/-0.000	1.161+/-0.001	0.473+/-0.000	-	0.477+/-0.000
Theta	0.105+/-0.000	**0.007+/-0.000**	1.083+/-0.000	0.558+/-0.000	0.331+/-0.000	0.622+/-0.000
TBATS	-	0.008+/-0.000	0.876+/-0.000	-	-	0.495+/-0.000
DeepAR	**0.056+/-0.001**	0.009+/-0.001	**0.426+/-0.004**	**0.289+/-0.001**	**0.117+/-0.001**	**0.226+/-0.002**
NPTS(uni.)	0.230+/-0.001	0.026+/-0.000	0.805+/-0.001	0.473+/-0.000	0.403+/-0.000	0.291+/-0.000
Seas.NPTS(uni.)	0.061+/-0.000	0.026+/-0.000	0.791+/-0.000	0.474+/-0.000	0.174+/-0.000	0.285+/-0.000
NPTS	0.230+/-0.001	0.021+/-0.000	0.802+/-0.001	0.473+/-0.000	0.403+/-0.000	0.277+/-0.000
Seas.NPTS	0.060+/-0.000	0.020+/-0.000	0.790+/-0.000	0.474+/-0.000	0.173+/-0.000	0.269+/-0.000
DeepNPTS	**0.059+/-0.004**	0.009+/-0.000	**0.447+/-0.011**	**0.413+/-0.015**	**0.163+/-0.036**	**0.232+/-0.000**

Table 7: Mean quantile losses averaged over 5 runs (lower is better). Best two methods are highlighted. The bottom half shows different variants of the NPTS method. `AutoARIMA` and `TBATS` exceeded the time limit of 24 hours for `Electricity` and `Traffic` datasets, so we do not report their numbers.

Additionally, to evaluate mean forecasts, we report the (normalized) root-mean squared errors (RMSE) for all the methods. Note that unlike mean quantile loss which evaluates the spread of

dataset	M4 daily	M4 hourly	M4 monthly	M4 quarterly	M4 weekly	M4 yearly
Seas.Naive	0.028+/-0.000	0.048+/-0.000	0.146+/-0.000	0.119+/-0.000	0.063+/-0.000	0.162+/-0.000
AutoETS	**0.022+/-0.000**	0.042+/-0.001	0.096+/-0.000	**0.076+/-0.000**	0.049+/-0.000	0.122+/-0.000
AutoARIMA	0.024+/-0.000	0.038+/-0.001	**0.094+/-0.000**	0.077+/-0.000	**0.048+/-0.000**	**0.120+/-0.000**
Theta	0.023+/-0.000	**0.041+/-0.000**	**0.094+/-0.000**	0.077+/-0.000	0.050+/-0.000	**0.116+/-0.000**
TBATS	**0.021+/-0.000**	**0.031+/-0.000**	-	**0.074+/-0.000**	**0.046+/-0.000**	0.123+/-0.000
DeepAR	**0.022+/-0.000**	0.050+/-0.003	0.112+/-0.002	0.086+/-0.004	0.051+/-0.004	0.130+/-0.007
NPTS(uni.)	0.161+/-0.000	0.115+/-0.001	0.257+/-0.000	0.289+/-0.000	0.319+/-0.001	0.389+/-0.000
Seas.NPTS(uni.)	0.161+/-0.000	0.053+/-0.000	0.258+/-0.000	0.288+/-0.000	0.329+/-0.001	0.389+/-0.000
NPTS	0.139+/-0.000	0.112+/-0.001	0.224+/-0.000	0.246+/-0.000	0.272+/-0.000	0.343+/-0.000
Seas.NPTS	0.139+/-0.000	0.046+/-0.000	0.225+/-0.000	0.245+/-0.000	0.285+/-0.000	0.343+/-0.000
DeepNPTS	0.027+/-0.000	0.065+/-0.019	0.149+/-0.012	0.097+/-0.002	0.057+/-0.001	0.168+/-0.008

Table 8: Mean quantile losses averaged over 5 runs (lower is better) for M4 datasets. Best two methods are highlighted. The bottom half shows different variants of the NPTS method. TBATS exceeded the time limit of 24 hours for M4 monthly dataset, so we do not report that number.

the forecast distribution, RMSE is a metric for point forecast and measures the error between the mean forecast $\hat{z}$ and the actual observation z. RMSE for a given dataset of N time series and T evaluation points is given by

$$\text{RMSE} = \sqrt{\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(z_{i,t} - \hat{z}_{i,t})^2}.$$

Since RMSE is a scale-dependent error, we normalize it by the total actual values for better readability. The normalized RMSE is given by

$$\text{NRMSE} = \frac{1}{\sum_{i=1}^{N}\sum_{t=1}^{T}|z_{it}|}\text{RMSE}.$$

The NRMSE scores are shown in Tables 9 and 10. Again, similar to the mean quantile loss, DeepNPTS consistently achieves good normalized RMSE values across the datasets and stands as one of top two methods for 4 out of 12 datasets.

dataset estimator	Electricity	Exchange Rate	Solar Energy	Taxi	Traffic	Wiki
Seas.Naive	**0.478+/-0.000**	0.016+/-0.000	1.368+/-0.000	**0.807+/-0.000**	0.613+/-0.000	3.052+/-0.000
AutoETS	1.393+/-0.023	**0.014+/-0.000**	2.153+/-0.016	1.198+/-0.001	0.646+/-0.002	6.025+/-0.053
AutoARIMA	-	**0.014+/-0.000**	1.974+/-0.002	0.969+/-0.001	-	3.059+/-0.004
Theta	0.812+/-0.000	**0.014+/-0.000**	2.030+/-0.000	1.161+/-0.000	0.636+/-0.000	2.797+/-0.000
TBATS	-	**0.014+/-0.000**	1.642+/-0.000	-	-	5.761+/-0.000
DeepAR	**0.711+/-0.034**	0.020+/-0.002	**1.119+/-0.005**	**0.603+/-0.002**	**0.406+/-0.003**	**2.123+/-0.006**
NPTS(uni.)	2.742+/-0.023	0.050+/-0.000	1.744+/-0.001	0.944+/-0.000	0.834+/-0.000	2.243+/-0.002
Seas.NPTS(uni.)	0.739+/-0.003	0.050+/-0.000	1.739+/-0.001	0.944+/-0.000	0.521+/-0.001	2.276+/-0.004
NPTS	2.752+/-0.013	0.041+/-0.000	1.740+/-0.002	0.944+/-0.000	0.834+/-0.000	2.229+/-0.003
Seas.NPTS	0.727+/-0.004	0.041+/-0.000	1.737+/-0.001	0.944+/-0.000	0.520+/-0.000	2.265+/-0.008
DeepNPTS	0.713+/-0.087	**0.014+/-0.000**	**1.217+/-0.025**	0.821+/-0.020	**0.490+/-0.066**	**2.180+/-0.000**

Table 9: Normalized RMSE values averaged over 5 runs (lower is better). Best two methods are highlighted. The bottom half shows different variants of the NPTS method. AutoARIMA and TBATS exceeded the time limit of 24 hours for Electricity and Traffic datasets, so we do not report their numbers.

A.2 Data Visualization

To illustrate the diversity of the datasets used, we plot the distribution of observed values in the training part of the time series marginalized over time and item dimensions (each dataset contains time series corresponding to different *items*). The plot is shown in Figure 3 for all the datasets.

dataset	M4 daily	M4 hourly	M4 monthly	M4 quarterly	M4 weekly	M4 yearly
Seas.Naive	0.109+/-0.000	0.260+/-0.000	0.339+/-0.000	0.264+/-0.000	0.123+/-0.000	0.324+/-0.000
AutoETS	**0.093+/-0.001**	0.293+/-0.006	0.294+/-0.000	**0.230+/-0.000**	0.119+/-0.000	0.332+/-0.000
AutoARIMA	0.099+/-0.000	0.307+/-0.010	**0.288+/-0.000**	0.240+/-0.001	**0.117+/-0.001**	0.331+/-0.000
Theta	0.097+/-0.000	**0.259+/-0.000**	**0.287+/-0.000**	**0.225+/-0.000**	0.121+/-0.000	**0.301+/-0.000**
TBATS	**0.095+/-0.000**	**0.210+/-0.000**	-	0.239+/-0.000	**0.118+/-0.000**	0.417+/-0.000
DeepAR	0.102+/-0.002	0.487+/-0.034	0.292+/-0.004	0.239+/-0.003	0.121+/-0.004	**0.302+/-0.005**
NPTS(uni.)	0.374+/-0.000	0.982+/-0.009	0.593+/-0.000	0.605+/-0.000	0.721+/-0.001	0.730+/-0.000
Seas.NPTS(uni.)	0.374+/-0.000	0.448+/-0.003	0.592+/-0.000	0.603+/-0.000	0.728+/-0.001	0.730+/-0.000
NPTS	0.340+/-0.000	0.957+/-0.004	0.544+/-0.000	0.546+/-0.000	0.647+/-0.001	0.675+/-0.000
Seas.NPTS	0.341+/-0.000	0.387+/-0.002	0.543+/-0.000	0.544+/-0.000	0.658+/-0.001	0.676+/-0.000
DeepNPTS	0.102+/-0.001	0.541+/-0.161	0.353+/-0.019	0.245+/-0.005	0.127+/-0.002	0.362+/-0.014

Table 10: Normalized RMSE values averaged over 5 runs (lower is better) for M4 datasets. Best two methods are highlighted. The bottom half shows different variants of the NPTS method. TBATS exceeded the time limit of 24 hours for M4 monthly dataset, so we do not report that number.

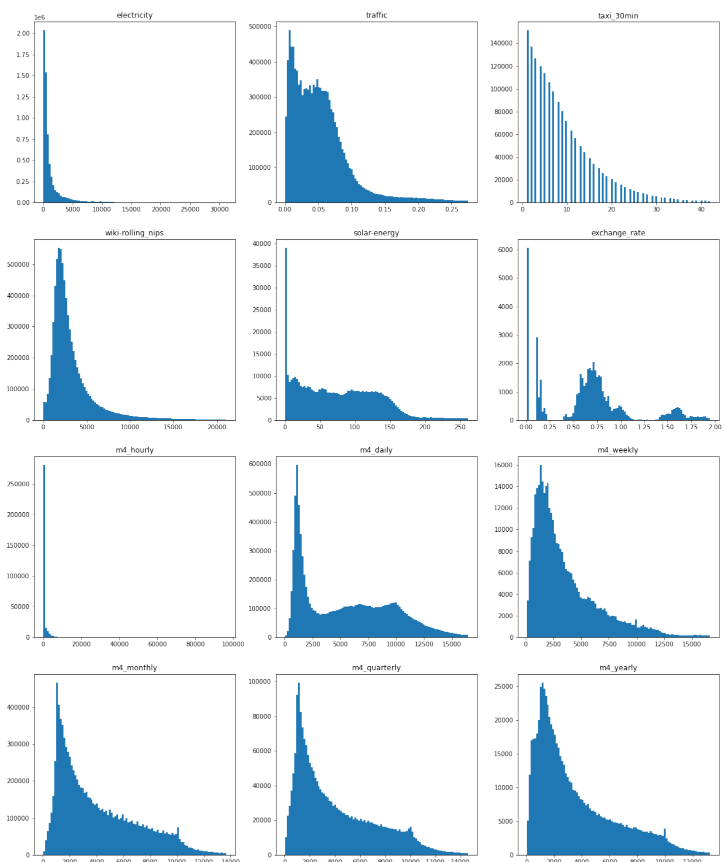


Figure 3: Histogram of observed values of the (training part of) time series for all datasets used in the evaluations.