Univariate Time Series Forecasting using ES-RNN and N-BEATS

Yves Greatti
New York University
yves.greatti@nyu.edu

Zian Chen
New York University
zian.chen@nyu.edu

Sunjoo Park
New York University
sunjoo.park@nyu.edu

Abstract

In this work, we seek to replicate and improve the results reached by two neural networks: **ES-RNN** [4] and **N-BEATS** [3] on the M4 dataset competition ([1]). We also run different experiments to compare the performances of these two deep learning techniques to a more classical statistical approach like ARIMA or Gaussian Process ($\mathcal{GP}$). We demonstrate that although Gaussian processes could be powerful for sampling tasks and simpler to configure, these neural networks outperform it for forecasting. Neural networks could have an overhead in term of the number of hyper-parameters to tune, but when using batching they scale up very easily and generalize well to a large number of time series (100K for M4). We are thus, pretty confident that, the two neural networks could forecast with the appropriate setting of hyper-parameters, other univariate time series beyond the M4 dataset.

1 Introduction

The dataset is provided by the M4 competition organized by the International Institute of Forecasters. These competitions also known as the "Makridakis competitions", have been happening since 1982, roughly every 10 years with an increasing number of time series to forecast starting from 1000 in 1982 to 100,000 in 2018, for the M4 competition. They attract people from academia as well practitioners, the last winner is Slawek Smyl, from Uber Technologies. The model created by Smyl is a hybrid approach combining Holts-Winter smoothing techniques with a recurrent neural network (RNN). Boris Oreshkin et al. [3] wanted to challenge the conclusion that, mixed techniques are the future by proposing a pure DL model with interpretable outputs: **N-Beats**, which they claimed outperforms **ES-RNN**, Smyl's model. N-Beats could be a very deep DL model and it is important for their authors that, in addition to hight accuracy, N-Beats has interpretable outputs which can match established statistical models such as ETS and ARIMA, robust, efficient, and automatic models.

2 Problem Description

Univariate point forecasting in discrete time consists in predicting future values given a series of observations.

- Given an history $[y_1, \cdots, y_T]$
- TASK: predict $[y_{T+1}, \cdots, y_{T+H}]$
- H: horizon, T: length of observations, m: periodicity of the data
- Standard scale-free metrics in the practice of forecasting:

[1] code available at `https://github.com/ygrepo/DS-GA-3001-001-Project`.

– sMAPE: symmetric Mean Absolute Percentage Error
– MASE: Mean Absolute Scaled Error

For the purpose of this research, we used the predictive accuracy metrics sMAPE and MASE which are the standard metrics across the forecasting community, disregarding OWA which is a metric defined by Makridakis and requires $\mathrm{MASE}_{\mathrm{Naive2}}$ as part of its computation, which is generated by a random walk model set up by the organizers of the competition.

$$\mathrm{sMAPE} = \frac{2}{H}\sum_{i=1}^{H}\frac{|y_{T+i} - \hat{y}_{T+i}|}{|y_{T+i}| + |\hat{y}_{T+i}|}\ ,\ \mathrm{MASE} = \frac{1}{H}\sum_{i=1}^{H}\frac{|y_{T+i} - \hat{y}_{T+i}|}{\frac{1}{T+H-m}\sum_{j=m+1}^{T+H}|y_j - y_{j-m}|}$$

2.1 Data Description

M4 dataset extends the previous three competitions by increasing significantly the number of series. The time series are grouped into six categories and can be split into two groups: high-frequency data (weekly, daily, hourly) along with low-frequency data (yearly, quarterly and monthly). They were built from real, multiple, and diverse sources and are divided into six data types: demographic, finance, industry, macro, micro and other. The minimum of observations for the training set are 79 for hourly, 652 for daily, 67 for weekly, 24 for monthly, 8 for quarterly and 7 for yearly series. They have different length ranging from 7 to about 1000 observations.

Frequency	**Demographic**	**Finance**	**Industry**	**Macro**	**Micro**	**Other**	**Total**
Yearly	1,088	6,519	3,716	3,903	6,538	1,236	23,000
Quarterly	1,858	5,305	4,637	5,315	6,020	865	24,000
Monthly	5,728	10,987	10,017	10,016	10,975	277	48,000
Weekly	24	164	6	41	112	12	359
Daily	10	1,559	422	127	1,476	633	4,227
Hourly	0	0	0	0	0	414	414
Total	8,708	24,534	18,798	19,402	25,1212	3,437	100,000

Table 1: M4 data by type and series frequency

	Mean	**Std-Dev**	**Min**	**25%**	**50%**	**75%**	**Max**
Yearly	25	24	7	14	23	34	829
Quarterly	84	51	8	54	80	107	858
Monthly	198	137	24	64	184	288	2776
Weekly	1009	707	67	366	921	1590	2584
Daily	2343	1756	79	309	2926	4183	9905
Hourly	805	127	652	652	912	912	912

Table 2: M4 series length statistics

3 ES-RNN

3.1 Model Description

ES-RNN algorithm consists in two major layers: a preprocessing layer which uses Holt-Winters smoothing technique to extract trend and seasonality parameters and a dilated RNN network. The Holt-Winter parameters are part of the back-propagation and are tuned as the network learns the characteristics of the time series. The ES-RNN model uses a modified version of the Holt-Winters formula, in which there is no local linear level coefficient, and level and seasonality terms are scaled instead of subtracted. The linear forecast is replaced by an RNN, which has for input the normalized and de-seasonalized prior observations.

$$\begin{aligned} l_t &= \alpha \frac{y_t}{s_{t-m}} + (1-\alpha) l_{t-1} \\ s_t &= \beta \frac{y_t}{l_{t-1}} + (1-\beta) s_{t-m} \\ \hat{y}_{\textbf{win}} &= \textbf{ES-RNN}(\frac{y_{ti}}{s_{ti} l_{ti}}) \\ y_{\textbf{truth}} &= (\frac{y_{to}}{s_{to} l_{to}}) \end{aligned}$$

where l is a state variable, s is a seasonality coefficient, α and β are network parameters, m is the periodicity of the data.

To model long-term dependencies in time series, ES-RNN uses dilated RNNs [1] which allow to stack RNN cells by having skip connections between RNN cells . The dilation parameters are given in table Table 3. DRNN cells can be vanilla RNN, LSTM, GRU, GRU cells provided for us the higher accuracy.

Time Series	Dilations
Quarterly	(1, 2), (4, 8)
Monthly	(1, 3), (6, 12)
Daily	(1, 7), (14, 28)
Yearly	(1, 2), (2, 6)
Weekly	(1, 14), (14, 28)
Hourly	(1, 24), (24, 48)

Table 3: ES-RNN dilation parameters

3.2 Data Preparation

The M4 time series have different length, to allow vectorization and to take advantage of the GPUs, they are chopped using a predetermined cut off value. To the exception of the daily time series, for which the cut-off value is 200, 72 is used as the default value (Table 4).

Filtering rates (in %)					
Hourly	**Daily**	**Weekly**	**Monthly**	**Quarterly**	**Yearly**
6.93	6.93	18.11	26.83	45.42	60.61

Table 4: Percentage of time series eliminated by the cut-off value

The batch size is by default 1024, we observe that, it has no major effects on the overall performance of the network.

For an hourly frequency, the effect of the batch size is given in table Table 5 .

Each time series is split into two windows: a backcast and forecast window. Observations in each window are normalized and de-seasonalized using the coefficients of the exponential smoothing (Figure 1 and Figure 2). In addition, a one-hot representation of the time series category is added to the input window.

	sMAPE per category						
Batch size	**Demographic**	**Finance**	**Industry**	**Macro**	**Micro**	**Other**	**Total**
512	6.32	3.34	3.95	2.52	2.43	3.08	3.02
1024	6.20	3.27	3.9	2.52	2.37	3.07	2.97
2048	6.26	3.32	3.92	2.49	2.42	3.1	3.01

Table 5: ES-RNN batch size statistics

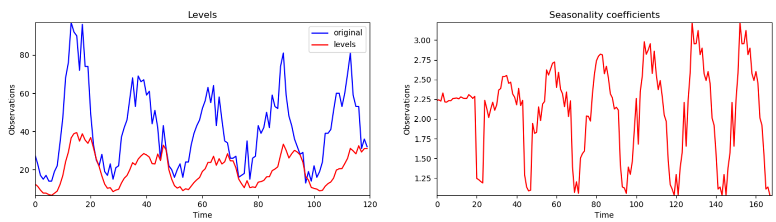


Figure 1: Levels are a smoothing version of the original time series and seasonality coefficients are between 0 and 3.

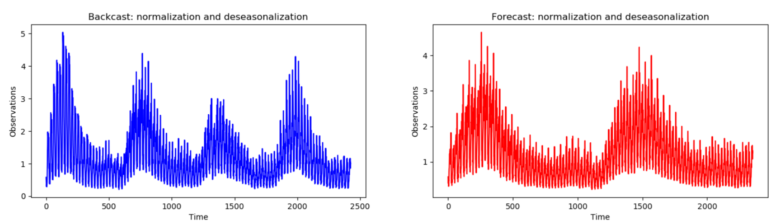


Figure 2: The input and ouput of the RNN are normalized and de-seasonalized, compared to N-BEATS which are not scaled.

3.3 Deep learning architecture

ES-RNN layers from top to bottom consist in a stack of dilated RNNs followed by a fully connected layer, an activation function and a dense layer (Table 6).

ES-RNN
time series $\in \mathbb{R}^{b \times l}$
RNN $\in \mathbb{R}^{l \times s}$
RNN $\in \mathbb{R}^{s \times s}$
$\ldots$
RNN $\in \mathbb{R}^{s \times s}$
dense $\in \mathbb{R}^{s \times s}$
Tanh
dense $\in \mathbb{R}^{s \times h}$

Table 6: ES-RNN architecture. Let b the batch size, l the length of the time series, s the embedding dimensional state size and h the length of the prediction.

Dropout or batch normalization did not provide any significant improvement, the smoothing and normalization of the data seemed enough for a convergence to a local minimum. In both networks, using the pinball loss [2] function brought definitely a boost in performance compared to L2 or L1 losses. The pinball loss function is popular in forecasting. Two different slopes are used depending whether the forecast $\hat{y}_t$ value is greater or smaller than the truth value y_t. Two models are then trained, corresponding to the lower and upper bounds of the desired level of prediction interval (Figure 3):

$$L_\alpha = \begin{cases} \alpha(\hat{y}_t - y_t) & \text{if } \hat{y}_t \geq y_t \\ (1-\alpha)(y_t - \hat{y}_t) & \text{if } y_t > \hat{y}_t \end{cases}$$

Then $\mathrm{P}(\hat{y}_{t,lower} \leq y_t < \hat{y}_{t,upper}) = 1 - 2\alpha$.

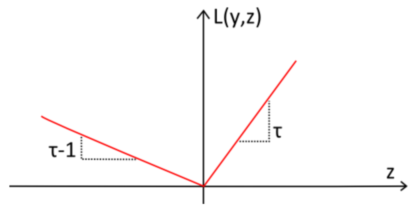


Figure 3: τ controls the desired imbalance in the quantile forecast.

3.4 Training of the network

Most of the parameters set by Smyl in his initial model were preserved and resulted in good performances. The only changes we made, was regarding the learning rate value and that we used a learning rate scheduler. All experiments for ES-RNN were made using Adam. We trained for 100 epochs starting with a learning rate of 0.01 which was progressively reduced to 1×10^{-5}. We also changed the batch size as detailed in the previous section and replaced the Tanh activation function with ReLu and LeakyReLu without noticeable improvements.

4 N-BEATS

4.1 Model Description

N-Beats architecture by design is simple and generic, yet expressive (deep) (Figure 4). It does not rely on any time-series specific feature engineering or input scaling. It is also extendable and the basic components are:

- **Block**

  A block accepts an input $\boldsymbol{x}$ which is a history lookback window, also known as "backcast". And the block outputs two vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{x}}$ is the block's best estimate of $\boldsymbol{x}$, $\hat{\boldsymbol{y}}$ is the block's forward forecast of length H. The length of the input window is a multiple of the forecast horizon H, and $\boldsymbol{x}$ value ranges from $2H$ to $7H$. Each block removes the portion of the signal it can explain well, making the forecast job of the downstream blocks easier. This also facilitates more fluid gradient backpropagation.

- **Stack**

  A stack is a collection of blocks and stacks can be hierarchically organized to make a very deep architecture. Each block provides a partial forecast, that is first aggregated at the stack level and then at the overall network level.

4.2 Additional model details

The input $\boldsymbol{x}$ goes first through four linear layers followed by two interpolation functions, one for backcast: g^b_θ and the other for forecast: g^f_θ (Table 7). There are three types of blocks:

generic : $\hat{\boldsymbol{y}}_{i,j} = \boldsymbol{W}_{i,j}\theta_{i,j} + \boldsymbol{b}_{i,j}$. This block outputs are not interpretable.

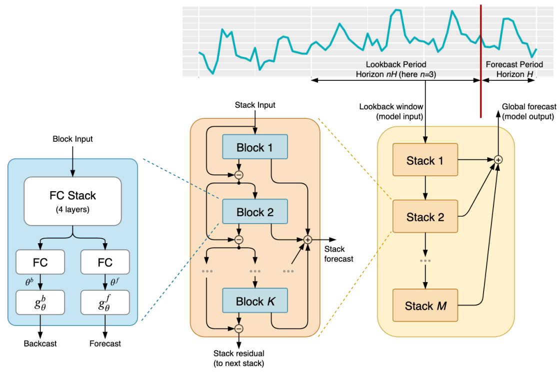


Figure 4: N-BEATS overall architecture.

trend model : the interpolation is a monotonic function, a polynomial of small degree p: $g_\theta = \sum_{i=0}^{p} \theta_i t^i$ where $\boldsymbol{t} = [0, 1, 2, \cdots, H-2, H-1]^T/H$ and p ≤ 4.
The trend forecast has the form:

$$\hat{\boldsymbol{y}}^{tr}_{i,j} = \boldsymbol{T}\theta_{i,j}$$

The number of θ parameters is less or equal to 4.

seasonality model : to capture cyclical, recurring patterns, this block performs a Fourier transform of the series:

$$g_\theta(t) = \sum_{i=0}^{\lfloor H/2-1 \rfloor} \theta_i \cos(2\pi i t) + \theta_{i+\lfloor H/2 \rfloor} \sin(2\pi i t) \text{ and p} < 10$$

where $\theta_{i,j}$ are Fourier coefficients predicted by a FC network of layer$j$ of stack i

$$\hat{\boldsymbol{y}}^{s}_{i,j} = \boldsymbol{S}\theta_{i,j}$$

The weight could be shared among the blocks within the same stack and resulted in better performance of the validation set.

N-BEATS
time series $\in \mathbb{R}^{b \times W \times bc}$
FC $\in \mathbb{R}^{bc \times s}$
ReLU
FC $\in \mathbb{R}^{bc \times s}$
ReLU
FC $\in \mathbb{R}^{s \times s}$
ReLU
FC $\in \mathbb{R}^{s \times s}$
ReLU
backcast $= g^b_\theta(x)$ forecast $= g^f_\theta(x)$

Table 7: N-BEATS architecture. Let b the batch size, W is the number of input windows, bc backcast length, s is the size of the hidden layer.

4.3 Training of the network

We split the data into training and validation and test datasets, similarly to what we did for ES-RNN model. We dynamically determined the cut-off value by using the minimal size of the time series which were larger than the minimum length: backcast + forecast. We ran various experiments with different configuration of the stacks, number of blocks per stack, size of the hidden layer, the batch size and values for the θ parameters. We also trained the network for 100 epochs with an initial learning rate of 0.01, which was progressively reduced by the scheduler to 1×10^{-7}.

The batch size is by default 1024, we observe that, it affects the overall performance of the network, depending the time series frequency, too small the validation loss jumped up and down, and with larger batch size, the performance was better. In addition, using dropout with a value of 0.2, helped to have a good accuracy.

Oreshkin et al. claimed that, their model is interpretable when looking at the outputs of the trend or seasonality blocks. We did not obtain exactly the same results (Figure 5), and this is maybe due to the fact we did not do ensembling or that we did not have a very deep architecture. They had 6 models for each time series category, a very deep network of total depth 150 layers, trained each model on input windows of different length: $2H, 3H, \cdots, 7H$ and followed a bagging procedure for 180 models.

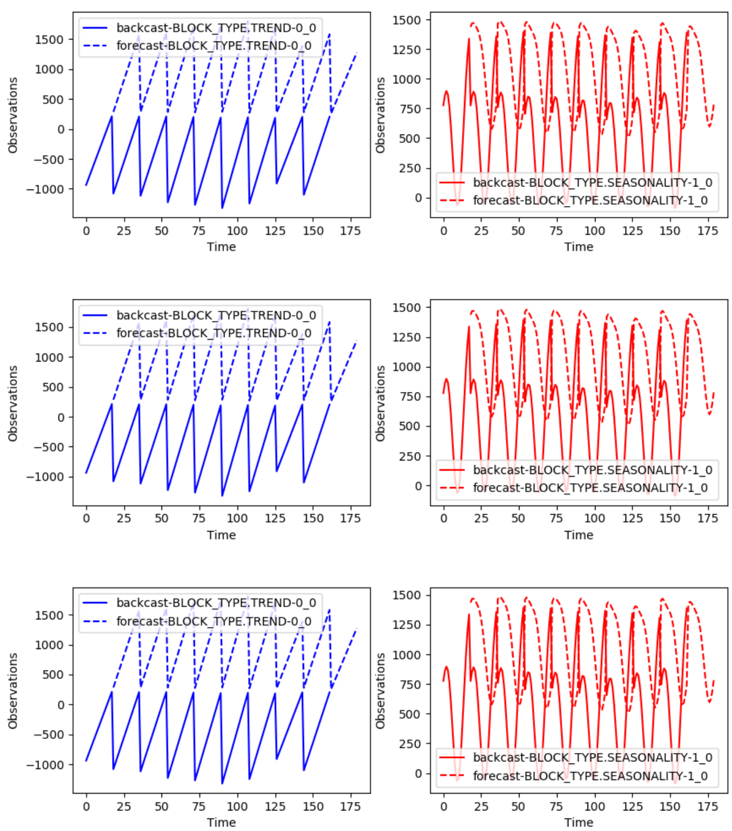
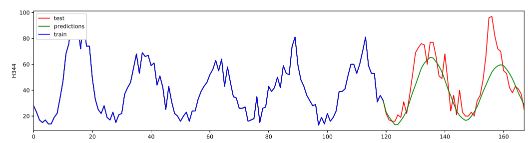


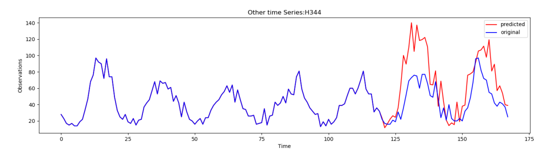
Figure 5: Trend and seasonal coefficients for the backcast and forecast outputs of the M1 time series.

5 Model comparison

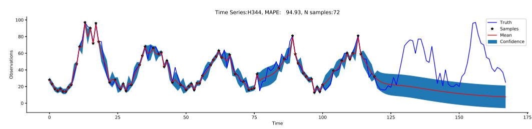
We used ARIMA and Gaussian Process to generate our baselines. ARIMA was one of the method used by the M4 competition organizers to generate the benchmarks. We also decided to use $\mathcal{GP}$ for their simplicity, the possibility to select different kernels and to have confidence intervals related to theirs forecasting. On a single time series, traditional statistical models like ARIMA were difficult to outperform, the downside is that they do not scale up with the number of time series, since each time series requires to carefully tune the ARIMA model (p, q, d) parameters. The Gaussian process performed the worst on for forecasting, maybe due to wrong choice of kernel. We verified visually the results using sMAPE scoring by plotting for specific time series the forecast of each model, as displayed in Figure 6 and Figure 7.



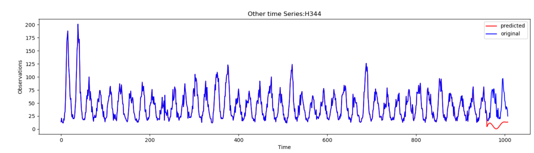
(a) ARIMA predictions



(b) ES-RNN predictions

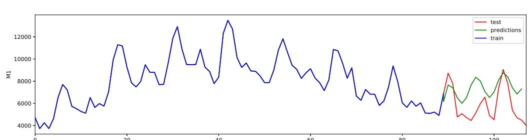


(c) GP predictions

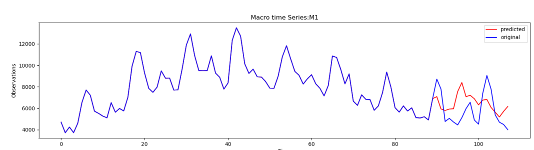


(d) N-BEATS predictions

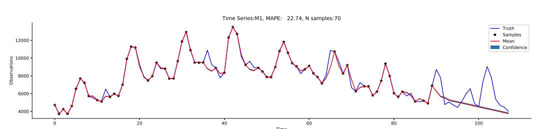
Figure 6: Predictions for H344 time series: ARIMA (top-left), ES-RNN (top-right), GP (bottom-left), N-BEATS (bottom-right)



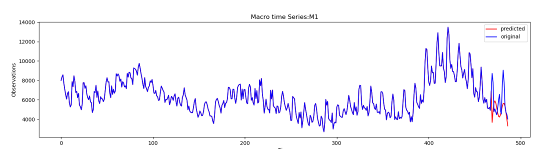
(a) ARIMA predictions



(b) ES-RNN predictions



(c) GP predictions



(d) N-BEATS predictions

Figure 7: Predictions for M1 time series: ARIMA (top-left), ES-RNN (top-right), GP (bottom-left), N-BEATS (bottom-right)

Overall ES-RNN seemed to perform better than N-BEATS which requires more tuning and to go deeper by increasing the number of layers like described in Table 8, Table 9, Table 10, and Table 11. In term of numbers parameters, ES-RNN used about 500,000 parameters and N-BEATS with an hidden size of 256 and one stack of two blocks (trend and seasonal) has about 400,000 parameters.

6 Conclusion

Pure DL models which provide interpretable results and do not require hand-crafted smoothing, are very powerful. Therefore neural networks are definitely the future for univariate forecasting. The question is still open whether the next generation of models will adopt an hybrid approach or not. Mean Squared error or L2 loss are ill-adapted to favor interesting

Model	sMAPE by Frequency H344	D1	W246	M1	Q66	Y1
ARIMA	**23.24**	**0.28**	**5.51**	30.40	5.96	11.20
GP	94.93	22.37	32.7	19.04	35.02	20
ES-RNN	36.68	6.48	23.44	**10.19**	10.2	**1.99**
N-BEATS	43.98	6.54	30.83	24.68	**1.47**	9.53

Table 8: Comparison of results on specific time series, in bold the best score, the lower the better.

Model	sMAPE by Frequency Hourly	Daily	Weekly	Monthly	Quarterly	Yearly
ES-RNN	30.26	**2.97**	14.84	**9.78**	10.51	14.58
N-BEATS	20.1	4.08	16.05	22.22	11.94	13.12

Table 9: Performance on the M4 test set, in bold when the models performed better than in the original papers.

Data Category	Yearly	Quarterly	Monthly
Demographic	**11.45**	11.67	**5.76**
Finance	**16.31**	**10.41**	**10.8**
Industry	21.98	8.74	**11.3**
Macro	**14.21**	10.13	**11.7**
Micro	**10.93**	11.96	**8.02**
Other	16.27	7.87	**7.76**
Overall	14.58	10.51	**9.78**

Table 10: ES-RNN: breakdown of sMAPE by time period and category, in bold when the models performed better than in the original paper.

Data Category	Yearly	Quarterly	Monthly
Demographic	10.34	11.74	5.57
Finance	14.11	13.90	15.36
Industry	17.16	10.85	14.76
Macro	13.39	11.35	14.6
Micro	10.92	12.22	15.2
Other	13.62	7.90	13.89
Overall	13.29	11.94	13.88

Table 11: N-BEATS: breakdown of sMAPE by time period and category

vs. naive forecasts. Our effort to wrap the sMAPE into a loss function were not conclusive. Nevertheless having an objective function, and in addition robust DL architectures, which take into account very long temporal dependencies, are probably the next focus of research in the domain of univariate forecasting.

References

[1] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael J. Witbrock, Mark Hasegawa-Johnson, and Thomas S. Huang. Dilated recurrent neural networks. *CoRR*, abs/1710.02224, 2017. URL `http://arxiv.org/abs/1710.02224`.

[2] Andreas Christmann Ingo Steinwart. Estimating conditional quantiles with the help of the pinball loss. *arXiv:1102.2101*, 2011.

[3] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. *CoRR*, abs/1905.10437, 2019. URL `http://arxiv.org/abs/1905.10437`.

[4] Andrew Redd, Kaung Khin, and Aldo Marini. Fast ES-RNN: A GPU implementation of the ES-RNN algorithm. *CoRR*, abs/1907.03329, 2019. URL `http://arxiv.org/abs/1907.03329`.