PES: *P*ROBABILISTIC *E*XPONENTIAL *S*MOOTHING FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series forecasting is a common task in many industries. It helps organizations in setting goals, making plans and taking decisions. Probabilistic forecasting, in addition, summarizes the confidence over future quantities, a useful property when targeting uncertainty. This paper proposes PES - *P*robabilistic *E*xponential *S*moothing -, a hybrid model for univariate time series forecasting. The contribution is two-fold: we introduce a RNN-like cell incorporating a simple exponential smoothing operator; building on this new cell we develop an intelligible and data-efficient model. The proposed solution shows several desirable characteristics; being easy to implement and fast to train, it can accommodate multiple seasonality and learn them via cross-learning. It can produce intervals as well as point-forecasts and its structure could be a valuable time series decomposition scheme. We test the PES model over a demand forecasting task on a well-known, publicly available, dataset. Finally we show that the results obtained compare favorably to the state-of-the-art.

1 INTRODUCTION

Being able to identify patterns and extract information from time series has been a central task since the early 1900s, when the first attempts in time series analysis had been carried forward. *M*oving *A*verages (MA) and *E*xponential *S*moothing (ES) were amongst the first techniques to be adopted, and extended, to solve the problem. *A*uto*r*egressive *M*oving *A*verage (ARMA), and their *I*ntegrated version (ARIMA), came up besides these methods in the early 1970s. All together they are prominent examples of the wider *S*tate *S*pace *M*odels (SSMs) class. Exponential Smoothing and AR(I)MA represent the standard *de facto* in the industry due to their intelligibility and ease of implementation. They indeed allow for structural assumptions - i.e. a physical interpretation -, present a narrow number of parameters - practically chosen beforehand, either through information criteria or by experience - and the associated training procedure is data efficient. On the other hand they are meant to be series-wise methods, unable to learn shared patterns and to scale gracefully with the time series corpora.

An alternative is offered by *N*eural *N*etworks (NNs), and especially their *D*eep *L*earning (DL) variant. Thanks to their capacity to extract features from raw data, DL models are able to identify complex patterns within - and possibly across - time series with less human intervention. However making fewer structural assumptions and being characterized by a massive number of parameters, Deep NNs' deduction process is difficult to explain and data eager to learn accurate models.

An interesting report published by McKinsey & Company (2020) shows organizations lagging in AI adoption. Amongst the respondents - representing the full range of regions, industries, company sizes, functional specialties, and tenures - only 50% of them had adopted AI in at least one business function, with high-tech and telecom companies leading the charge. Within this fifty per cent, only 16% pushed a DL solution beyond the piloting stage. One of the relevant risk perceived by organizations in adopting AI is the lack of *explainability* - i.e. the ability to explain how AI models come to their decisions - and by extension a lack of trust in AI's judgement. A concern already raised by Makridakis et al. (2018) whom discuss the comparison discrepancy between research's advances and standard - or even naïve - methods, negatively impacting the industrial eligibility of proposed ML solutions.

In this paper we address both the explainability problem and the time series forecasting, blending standard ES methods and ML approaches. To the best of our knowledge, we present a new kind of RNN cell, direct translation of the common *S*imple *E*xponential *S*moothing (SES) method. In addition we develop a new model grounded on the recently introduced cell. The model retains the intelligibility and data efficiency of the SES method, while being able to learn multiple seasonal patterns across related raw time series and relative temporal co-variates. It adapts efficiently to the dataset's dimension, scaling smoothly from small to large corpora.

The remainder of the paper is organized as follow. In Section 2 we discuss the related works; Section 3 present the time series forecasting problem together with a brief review of the SES method. Section 4 summarises the proposed model, finally Section 5 present the experimental results.

2 RELATED WORKS

A good portion of the time series forecasting literature is extensively covered in (Durbin & Koopman, 2001; De Gooijer & Hyndman, 2006; Hyndman et al., 2008a; Hyndman & Athanasopoulos, 2021). Alongside Exponential Smoothing (Pegels, 1969), AR(I)MA (Yule, 1927; Granger & Joyeux, 1980) and SSMs (Kalman, 1960; Harvey, 1990; Durbin & Koopman, 2001) we can enumerate other classical methods, e.g. Croston (Croston, 1972). Harvey (1990) introduced the notion of *structutal time series*, decomposing a series into building blocks with associated physical interpretation. The main three are: *level*, *trend* and *seasonality*.

The Exponential Smoothing family can be condensed in three different methods from which several variations had been derived over the years. All of them assign to the observed values a proxy role, a mean to measure the *latent state* controlling the underneath smoothing process(es). The first, and most basic, method is the aforementioned simple exponential smoothing. It had been followed by the Holt method (Holt, 1957) and the Holt-Winter method (Winters, 1960) which respectively make room for trend and seasonality.

Progressively ML and DL models came up beside the standards. An unusual literature review can be achieved following the Makridakis Competition - or M Competition -, arrived at its fifth edition. Since its introduction the M Competition defined a common ground for researchers and practitioners to benchmark their forecasting approaches, adopting real-world datasets and metrics. The key conclusion of the first three editions was a benefit in creating an ensemble of different models, versus exploitation of single models, but the gain was not systematic. The winner of the M3 Competition had been the Theta model (Assimakopoulos & Nikolopoulos, 2000), Hyndman & Billah (2003) had later proven that the model was a variation of SES with drift. The M4 Competition had been the first one in the series asking the competitors to produce prediction intervals and hosting ML solutions. Smyl (2020) won the competition with a hybridization between ES methods and *R*ecurrent *N*eural *N*etworks (RNNs), specifically a neural residual dilated/attention *L*ong *S*hort-*t*erm *M*emory (LSTM) stack. To be precise a stack of LSTMs with different residual/dilation/attention had been provided for each horizon requested, plus one specific for prediction intervals generation. The seasonality had been handcrafted, letting the model accommodate at most two fixed seasonality. The solution outperformed all the benchmarks and strengthened the idea that a combination of models is fitted for the time series forecasting task. Finally the M5 Competition highlighted the relevance of cross-learning; involving retail time series, learning across related time series was the focal strategy to extract useful patterns. Retail time series, as well as other socio-economic series, are *count* data - i.e. made by integer non-negative observations - and can be *intermittent*, i.e. sporadic positive values are followed by long runs of zeroes. The M5 Competition also revealed the reluctance of the community in being involved with predictive distributions and uncertainty. The main event indeed was actually split in two sub-events, one named *Accuracy* (Makridakis et al., 2022a) and the other called *Uncertainty* (Makridakis et al., 2022b) following the main tasks. While the Accuracy part attracted more than 4'000 teams and counted over 99'000 submissions, its counterpart received much less attention with around 1/5th of competitors (892 teams) and 1/10th of submissions.

Similar solutions to Smyl (2020) - i.e. based on RNNs or hybrid between RNNs and other models - and able to learn across multiple time series had been presented by Salinas et al. (2019) (*DeepAR*) and Rangapuram et al. (2018) (*DeepSSMs*). Both of them treat the time series forecasting problem via probabilistic forecasting, i.e. generating a predictive distribution instead of a collection of *point forecasts* - a single value for each time step -, but with a different choice of the target dis-

tribution. Salinas et al. (2019) train a global RNN with a *Negative Binomial* as target distribution. The parameters/factors estimated by the global network are then passed to all the related time series. Rangapuram et al. (2018) propose a similar construction but differ in the execution. The patterns extracted by the global RNN are mapped to singular SSMs at series level, hence each time series receive the same global information but can form its own parameters. The target distribution is the classical *Gaussian*, useful to better exploit the Kalman filtering (Kalman, 1960) at the base of SSMs.

3 BACKGROUND

3.1 PROBLEM STATEMENT

Let $\mathcal{D} = \left\{\boldsymbol{z}^i_{t_{0,i}:T_i}\right\}_{i=1}^N$ be a dataset of N univariate time series. Each time series is a collection of observations such that $\boldsymbol{z}^i_{t_{0,i}:T_i} = \left\{z^i_{t_{0,i}}, \dots, z^i_t, \dots, z^i_{T_i}\right\}$, with $z^i_t \in \mathbb{R}$. The objective is to predict the future values $\left\{\boldsymbol{z}^i_{T_i+1:T_i+h}\right\}_{i=1}^N$, being h the forecasting horizon. Forecast values for the i-th series are denoted as $\hat{\boldsymbol{z}}^i_{T_i+1:T_i+h}$. An additional set of co-variates $\left\{\boldsymbol{x}^i_{t_{0,i}:T_i+h}\right\}_{i=1}^N$ can be given. The multi-step ahead time series forecasting can be stated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \sum_{t=T_i+1}^{T_i+h} \mathcal{L}(z^i_t, \hat{z}^i_t) \\ & \hat{z}^i_{T_i+1:T_i+h} = f(\boldsymbol{z}^i_{t_{0,i}:T_i}, \boldsymbol{x}^i_{t_{0,i}:T_i+h}), \end{aligned} \tag{1}$$

where $\mathcal{L}(z, \hat{z})$ is the loss function of choice. When $h = 1$ the multi-step ahead forecasting problem collapses to the one-step-ahead one. Multi-step and one-step-ahead forecasting are both instances of point forecast.

Probabilistic forecasting is an extension to the problem stated in equation 1; rather than emitting a single value, a probability is assigned to each element of a set of plausible values. The set of probabilities define the probability forecast. However, assigning a probability distribution to each time step could be unfeasible, hence in practice it is preferable to assign a probability distribution to the horizon of interest. Equation 1 is modified accordingly

$$\begin{aligned} \min \quad & \sum_{N} \sum_{t=T_i+1}^{T_i+h} \mathcal{L}(z^i_t, \hat{z}^i_t) \\ & P\big(\hat{\boldsymbol{z}}^i_{T_i+1:T_i+h} \big| \boldsymbol{z}^i_{t_{0,i}:T_i}, \boldsymbol{x}^i_{t_{0,i}:T_i+h}\big). \end{aligned} \tag{2}$$

The first advantage is that we can directly asses the goodness of our model inspecting the confidence intervals: the narrower a confidence interval is, the more the model has a higher confidence over its prediction. If we try to summarize the obtained probability distribution via a single number, mainly the expected value, we fall back to the point forecast case. The preferred loss function for this task is the *(log-)likelihood*. The (log-)likelihood should be chosen wisely to match the statistical properties of the data. Dealing with strictly positive observations, common modeling choices are either the Poisson distribution or the Negative Binomial one. Other choices are possible but these two are the most used distributions in practice and widely reviewed in literature.

3.2 EXPONENTIAL SMOOTHING METHODS

Simple exponential smoothing is fully described by its latent state $\boldsymbol{\eta}_t = [l_t]$ and the *smoothing parameter* α; $0 \leq \alpha \leq 1$. The state component l_t is called level. At time t the predicted value is found via

$$\begin{aligned} l_t &= \alpha z_{t-1} + (1-\alpha) l_{t-1}, \\ \hat{z}_t &= l_t. \end{aligned} \tag{3}$$

Holt (Holt, 1957) extended the SES with a trend equation. The new component comes with a smoothing parameter $0 \leq \beta \leq 1$ and a state component b_t, such that $\boldsymbol{\eta}_t = [l_t, b_t]$. We group

together the smoothing parameters under the smoothing vector $\boldsymbol{\phi} = [\alpha, \beta]$. Equation 4 shows the extension.

$$\begin{aligned} l_t &= \alpha z_{t-1} + (1-\alpha)(l_{t-1} + b_{t-1}), \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \\ \hat{z}_t &= l_t + hb_t. \end{aligned} \tag{4}$$

Finally the Holt-Winters method (Winters, 1960) schema is defined in equation 5. The state vector as well as the smoothing vector are updated accordingly: $\boldsymbol{\eta}_t = [l_t, b_t, s_t]$; $\boldsymbol{\phi} = [\alpha, \beta, \gamma]$, with $0 \leq \gamma \leq 1$.

$$\begin{aligned} l_t &= \alpha \frac{z_{t-1}}{s_{t-(1+m)}} + (1-\alpha)(l_{t-1} + b_{t-1}), \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \\ s_t &= \gamma \frac{z_{t-1}}{(l_{t-1} + b_{t-1})} + (1-\gamma)s_{t-(1+m)}, \\ \hat{z}_t &= (l_t + hb_t)s_{t-m+h_m^+}, \end{aligned} \tag{5}$$

Here m refers to the seasonal period - e.g. $m = 12$ for monthly, $m = 7$ for weekly - and $h_m^+ = (h-1) mod\ m$. In equation 4 and equation 5 we implicitly used an additive trend and a multiplicative seasonality. Please refer to Hyndman et al. (2004), Hyndman et al. (2002) and Hyndman & Athanasopoulos (2021) for a complete taxonomy.

3.3 METRICS

Common metrics employed to evaluate model performances, are *R*oot *M*ean *S*quared *E*rror (RMSE), *M*ean *A*bsolute *P*ercentage *E*rror (MAPE), *s*ymetric MAPE (sMAPE) and *M*ean *A*bsolute *S*caled *E*rror (MASE). The RMSE is scale-dependent, hence can not be used to compare different forecasting models. Mean absolute percentage error, sMAPE and MASE are instead scale-free error hence suitable for a comparison; however they have their own pitfalls. All of them are undefined if the one of the observed values - or both the observation and the forecast - happens to be zero. To prevent such event, for example, the M4 Competition's organizers treated the raw data to guarantee non-negative lower-bounded values with a lower threshold at 10. Moreover for strictly positive data, MAPE and sMAPE treat under-focasting and over-forcasting differently, penalizing more the latter case. This means that MAPE and sMAPE can be lower for biased than unbiased forecast.

When data are asymmetric - e.g. retail series - a different metric is more advisable. Following Seeger et al. (2016), Salinas et al. (2019) and Rangapuram et al. (2018) we employ the ρ-risk metric, a modified version of the Pinball Loss. The metric is defined as

$$L_\rho(z, \hat{z}) = \begin{cases} \rho(z - \hat{z}) & z \geq \hat{z} \\ (1-\rho)(\hat{z} - z) & z < \hat{z} \end{cases} \quad \rho \in (0, 1)$$

or compactly

$$L_\rho(z, \hat{z}) = I_{z \geq \hat{z}} \rho |z - \hat{z}| + I_{z < \hat{z}}(1-\rho)|z - \hat{z}|.^1 \tag{6}$$

Assigning different values to the ρ parameter, different weights are assigned to under- and over-forecasting. Specifically the value of ρ refers to a quantile of interest; when $\rho = 0.5$ - later referenced as $p50$ in section 5 - the loss reduces to *M*ean *A*bsolute *E*rror (MAE) up to a constant factor.

4 PROBABILISTIC EXPONENTIAL SMOOTHING

4.1 THE SIMPLE EXPONENTIAL SMOOTHING CELL

We start, for explanation's sake, from equation 5. First of all, the trend component has been discarded; it is common to experience trend when related time series are aggregated together, or in the

[1] $I_\star$ is and indexing function assessing on which side the forecast lies.

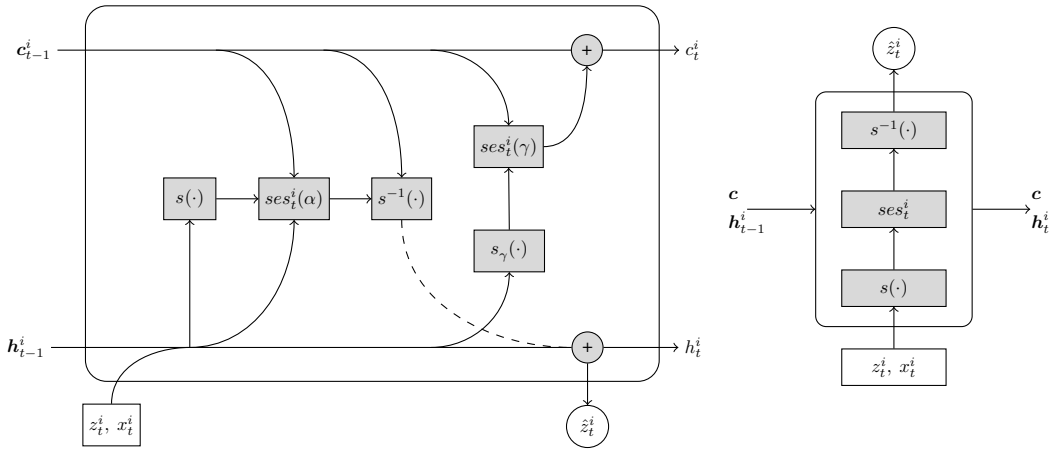

Figure 1: *(Left)*Single series mode: both level and seasonality are defined series-wise. Trend has been dropped. Scaling and re-scaling operations have been decoupled from the smoothing processes to isolate several operative blocks, allowing an independent implementation. This also makes room for multiple seasonality. The state vector is $\boldsymbol{h}^i_t = [z^i_t, l^i_t]$; the context vector $\boldsymbol{c}^i_t$ contains seasonal factors and their dilated version. *(Right)*Global mode: sharing the seasonality across different related time series, the seasonal blocks can be removed from the cell; the context vector gets a fixed dimension, multiple of the number of seasonlity and their period.

form of a *cycle* - a kind of trend appreciated on a long time span, typically decades or more -, but it is less usual than expected at single series level. Equation 5 subsides to equation 7

$$
\begin{aligned}
l^i_t &= \alpha \frac{z^i_{t-1}}{s^i_{t-(1+m)}} + (1-\alpha) l^i_{t-1}, \\
s^i_t &= \gamma \frac{z^i_{t-1}}{l^i_{t-1}} + (1-\gamma) s^i_{t-(1+m)}, \\
\hat{z}^i_t &= l^i_t \; s^i_{t-m+h^+_m}.
\end{aligned} \tag{7}
$$

Exponential Smoothing methods are iterative by design, we can surely identify the previous state $\boldsymbol{h}^i_{t-1} = [z^i_{t-1}, \boldsymbol{\eta}^i_{t-1}] = [z^i_{t-1}, l^i_{t-1}]$ and outline the next state $\boldsymbol{h}^i_t = [z^i_t, l^i_t]$. The seasonal factors $s^i_{t-(1+m)}$ and $s^i_{t-m+h^+_m}$ represent a sort of "attention" mechanism, forwarding information from past states to the actual one. They can be gathered into the context vector $\boldsymbol{c}^i_{t-1}$. The inner smoothing processes contain also three sub-operations: two scaling procedures - the scaling of the input observation in both the level equation and the seasonal equation - plus a re-scaling before emitting the forecast. Figure 1 illustrates how the different elements highlighted are translated to operative blocks of an RNN cell; the *SES cell*. The $\oplus$ operator defines an update of either the state vector or the context one. If seasonality is defined at single series level, the context vector has to be large enough to hold both the seasonal factors and their dilated version, e.g. $\boldsymbol{c}^i_{t-1} = [s^i_{t-M}, \dots, s^i_{t+M}]$ with $M = 2\max(m, h)$. Treating seasonality in a global fashion, we drop the series index - together with the relative blocks within the cell - and fix the context vector length such that $\boldsymbol{c} = [s_1, \dots, s_m]$. Having decoupled the scaling/re-scaling operators from the smoothing processes we made room for multiple seasonality; indeed we can implement the scaling blocks independently and easily extend the context vector $\boldsymbol{c} = [s^{g_1}_1, \dots, s^{g_1}_{m_1}, \dots, s^{g_{n_g}}_1, \dots, s^{g_{n_g}}_{m_{n_g}}]$. Elements $s^g_{1:m_g}$ identify the seasonal factors for the g-th granularity exposing the period m_g; n_g is the number of seasonality we wish to model. The reworked cell for the global case is shown in figure 1.

The schema presented here is not bounded to the use-case discussed, being easily generalized to all the ES variants within the taxonomy. A similar approach is found in Smyl (2020) where the Holt-Winter method is employed to pre-process the data to be fed to the RNN. In our case instead the Holt-Winter method is fused inside the RNN cell.

4.2 TRAINING

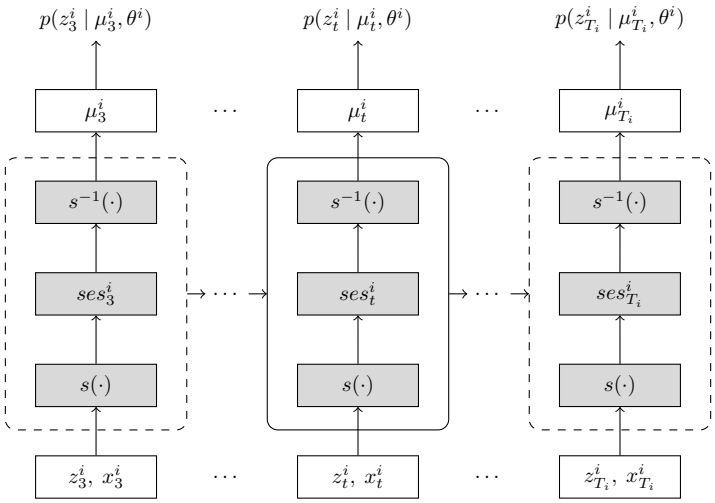

Figure 2: Summary of the model. Observations, co-variates and seasonal factors are passed to the scaling operator $s(\cdot)$ to produce seasonal-free time series. The latter are passed to the smoothing block ses_t^i. Seasonal effects are applied back by mean of $s^{-1}(\cdot)$. Time steps $t = 1, 2$ are omitted, being solely "warm-up" steps.

Our Probabilistic Exponential Smoothing (PES) model is schematized in figure 2, with the SES cell in global mode, focusing on solving the probabilistic forecasting problem for univariate time series. Co-variates $\boldsymbol{x}$ are assumed to be known over the complete time range $\{t_{0,i}, \dots, T_{i+h}\}$ and defined per-series, i.e. $\boldsymbol{x}^i_{t_{0,i}:T_{i+h}}$. The seasonal factors $\boldsymbol{s}^g_{1:m_g}$ are defined for each granularity of interest, shared and learnt jointly across all time series. A smoothing factor α^i is assigned to each time series in $\mathcal{D}$ such that $\boldsymbol{\phi} = [\alpha^1, \dots, \alpha^N]$.

Likelihood The Negative Binomial distribution has been chosen to train PES. Its probability mass function is defined as

$$P(z \mid r, p) = \frac{\Gamma(z + r)}{z!\Gamma(r)}(1 - p)^z p^r$$

where r is the number of failures, p the success rate - together they fully describe the distribution - and $\Gamma(\cdot)$ is the Gamma function. Knowing r and p is possible to compute the mean $\mu = {}^{pr}/_{(1-p)}$ and the variance $\sigma^2 = {}^{pr}/_{(1-p)^2}$. Other formulations are possible and available in literature, one frequently used substitutes the parameter r with its inverse ${}^1/_r$. We found effective in practice the notation introduced by Snyder et al. (2012), based on the time-varying mean μ_t and the parameter θ such that:

$$r_t = \frac{\mu_t}{\theta}; \qquad p = \frac{\theta}{1 + \theta};$$

$$p(z_t \mid \mu_t, \theta) = \frac{\Gamma(z_t + {}^{\mu_t}/_{\theta})}{z!\Gamma({}^{\mu_t}/_{\theta})} \left(\frac{1}{1 + \theta}\right)^z \left(\frac{\theta}{1 + \theta}\right)^{{}^{\mu_t}/_{\theta}}. \tag{8}$$

Our model is fully specified by the set of parameters

$$\boldsymbol{\psi} = [\boldsymbol{s}^{g_1}_{1:m_1}, \dots, \boldsymbol{s}^{g_{n_g}}_{1:m_{n_g}}, \boldsymbol{\phi}, \boldsymbol{\theta}] \tag{9}$$

where $\boldsymbol{\theta} = [\theta^1, \dots, \theta^N]$. Given the dataset $\mathcal{D}$, the model parameters are learnt maximizing the log-likelihood function, i.e.

$$\boldsymbol{\psi}^\star = \arg\max_{\boldsymbol{\psi}} \ \mathcal{L}(\boldsymbol{\psi})$$

$$\mathcal{L}(\boldsymbol{\psi}) = \sum_1^N \log P\big(\mathbf{z}^i_{1:T_i} \ \big|\ \mu^i_{1:T_i}, \theta^i\big). \tag{10}$$

The model distribution consists of a product of likelihood factors

$$P\big(\boldsymbol{z}^i_{T_{i+1}:T_{i+h}} \mid \boldsymbol{z}^i_{t_{0,i}:T_i}, \boldsymbol{x}^i_{t_{0,i}:T_{i+h}}; \boldsymbol{\psi}\big) = \prod_{t=T_{i+1}}^{T_{i+h}} P\big(z^i_t \mid \boldsymbol{z}^i_{t_{0,i}:T_i}, \boldsymbol{x}^i_{t_{0,i}:T_{i+h}}; \boldsymbol{\psi}\big)$$
$$= \prod_{t=T_{i+1}}^{T_{i+h}} P\big(z^i_t \mid \boldsymbol{\mu}^i_{t_{0,i}:T_i}, \theta^i\big). \tag{11}$$

Proceeding as depicted in figure 2 the observations $\boldsymbol{z}^i_{t_{0,i}:T_i}$ are scaled via $s(\cdot)$, applying the factors $[\boldsymbol{s}^{g_1}_{1:m_1}, \ldots, \boldsymbol{s}^{g_{n_g}}_{1:m_{n_g}}]$

$$\bar{\boldsymbol{z}}^i_{t_{0,i}:T_i} = s(\boldsymbol{z}^i_{t_{0,i}:T_i}, \boldsymbol{c}, \boldsymbol{x}^i_{t_{0,i}:T_i}).$$

These new seasonal-free values are passed to the smoothing block in place of the original observations. The smoothing block outputs the quantities $\tilde{\boldsymbol{z}}^i_{t_{0,i}:T_i} = ses(\bar{\boldsymbol{z}}^i_{t_{0,i}:T_i}, \boldsymbol{c}, \boldsymbol{h})$. The time-varying mean $\boldsymbol{\mu}^i_{t_{0,i}:T_i}$ is finally computed inverting the scaling function $s(\cdot)$

$$\boldsymbol{\mu}^i_{t_{0,i}:T_i} = s^{-1}(\tilde{\boldsymbol{z}}^i_{t_{0,i}:T_i}, \boldsymbol{c}, \boldsymbol{x}^i_{t_{0,i}:T_i});$$

The $\boldsymbol{\mu}^i_{t_{0,i}:T_i}$ and the parameter θ^i are then used to compute the likelihood $P(\boldsymbol{z}^i_{t_{0,i}:T_i} \mid \boldsymbol{\mu}^i_{t_{0,i}:T_i}, \theta^i)$.

4.3 PREDICTION

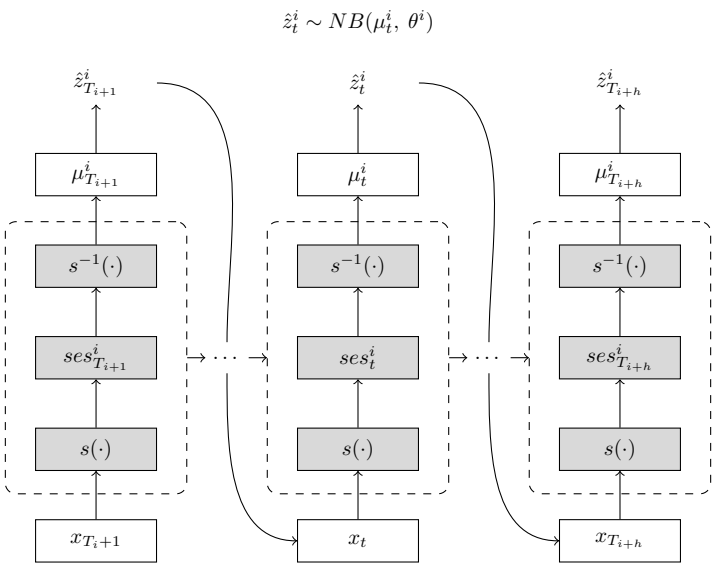

Figure 3: Illustration of the sampling process. For each time step $t \in [T_{i+1}, \ldots, T_{i+h}]$ a new sample $\hat{z}^i_t$ is drawn from the Negative Binomial distribution characterized by μ^i_t and θ^i; the smoothing process proceeds, producing a new mean μ^i_{t+1} which is further used to generate the next input, and so on until the whole time range is covered. The process is repeated multiple times to better approximate the true conditional distribution.

Prediction of future values is achieved via Monte Carlo simulation. For each time series we start from the $\mu^i_{T_i}$, generating a new sample

$$\hat{z}^i_{T_i} \sim NB(\mu^i_{T_i},\, \theta^i)$$

and passing it to the cell. The new mean $\mu^i_{T_i+1}$ is computed and a new sample is ready for the successive step. The process - depicted in figure 3 - is repeated over the whole prediction horizon, for all the time series, until K trajectories are generated for each of them. The mean value, computed for each time step, will represent the point forecast of the model; from the approximated distribution we can derive quantiles for the interval prediction.

5 EXPERIMENTAL RESULTS

Our model is fully defined and trained in Julia (Bezanson et al., 2012; Rackauckas et al., 2019), an emerging programming language in the data science community. Our experimental results are based on the ρ-risk metric introduced in section 3.3. We compare against Salinas et al. (2019) and Rangapuram et al. (2018), challenging them in a demand forecasting task over the *parts* dataset. We had specifically chosen these two models because their implementation is available both open-source - GluonTS (Alexandrov et al., 2019) - and commercially in Amazon's SageMaker. Indeed both solutions came from Amazon Research Labs. The dataset is publicly avaible.

5.1 PARTS DATASET

Presented in Hyndman et al. (2008a), the dataset comprises 2674 monthly series of slow moving parts supplied by a US car company. Covering a period of 51 months - from January 1998 to March 2002 - the majority of the time series, 89% of them specifically, are over-dispersed with an average dispersion ratio of 2.3. Only 2059 time series possess a complete history - i.e. no missing data - with an average gap between positive demands of 2.9 months. To avoid any computational problem with time series containing too few positive non-zero observations, the number of time series had been culled from 2059 to 1046 keeping only those time series with: a) at least 10 months with positive demand; b) at least some positive demands in the first 15 and in the final 15 months. The dataset had been used by Snyder et al. (2012), Salinas et al. (2019) and Rangapuram et al. (2018). To be aligned with Rangapuram et al. (2018) the dataset had been splitted into a training and a testing set. The training set contains the first 39 months - i.e. $T_i = 39$ - while the last year is used as prediction range, i.e. $T_{i+h} = 51$ with $h = 12$. The initial time step is equal for all the time series, i.e. $t_{0,1} = t_{0,2} = \cdots = t_{0,N} = 1$. The original dataset is available in its raw format at Hyndman et al. (2008b).

5.2 RESULTS

Since the data is monthly, we used only *M*onth-*o*f-*Y*ear (moy) indexes as co-variates, i.e. $\boldsymbol{x}^i_{1:51} = [1, 2, \ldots, 12, \ldots, 6, \ldots 3]$ since the time span starts in January and ends in March. The only granularity contained into the context $\boldsymbol{c}$ is $\boldsymbol{s}^{moy}_{1:12}$; the factors had been initialised uniformly. The parameters α^i are all initialized to the constant value 0.88, common choice in practice. The training lasted for 30 epochs with a learning rate of $5e{-}2$. No particular tuning procedure had been put in place. At prediction time we generated $K = 200$ trajectory, following the examples in Salinas et al. (2019) and Rangapuram et al. (2018) for the final distribution approximation. Table 1 presents the final ρ-risk. Our model, even if simple, is able to achieve a better $p50$ than DeepSSMs and a better $p90$

Table 1: Accuracy metrics on the *parts* dataset over the time span. **Bold** - best performing model. Red - second best performing. *auto.arima* is implemented in R's *forecast*

	auto.arima	DeepAR	DeepSSMs	PES
$p50$	1.6444	**1.273**	1.47	1.3539
$p90$	1.0664	1.086	**0.935**	1.0273

than DeepAR, being competitive with more complex solutions. A forecasting example is shown in figure 5 in Appendix A.1

5.3 EXPLAINABILITY

By design PES model can be inspected, as it is shown in figure 1. Taking the parts dataset as an example, we can easily inspect the seasonal profiles, which are presented in figure 4. This is a great advantage from the industrial point-of-view; thinking about the report by McKinsey & Company (2020) such a level of transparency will answer to practitioners' requirement of understanding, involving poor or no knowledge at all of complex ML structures. Explainability in this context should not be confused with explainable AI, rather it refers to the flexibility of a model to accept domain-specific constraints in order to be inherently interpretable, as perfectly outlined by Rudin (2019). On the contrary explainable AI is inclined to put in place an auxiliary model to explain the main one.

Approaches like *SH*apley *A*dditive ex*P*lanations (SHAP) and *L*ocal *I*nterpretable *M*odel-agnostic *E*xplanations (LIME) are frequently cited as mean to interpret deep models' outcome. However it has been proven that they can be fooled, as in Slack et al. (2020) work. Other ways in which auxialiary models can be deceived are explained in Rudin (2019).

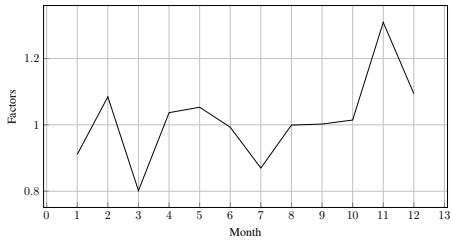

Figure 4: Monthly scaling factors learnt jointly across *parts* dataset time series.

6 CONCLUSIONS

In this paper we proposed the Probabilistic Exponential Smoothing (PES) model, a hybrid-model for univariate time series in the context of probabilistic forecasting. The model is grounded on a new kind of RNN cell, the SES cell, which lies a bridge between the well known, and widely used, statistical family of the Exponential Smoothing methods and the Neural Networks. The cell is a direct translation of the Simple Exponential Smoothing method after several operative building blocks, together with a state vector and a context vector, have been identified and implemented. We have shown the cell's derivation in the assumption of a multiplicative seasonality, explaining how multiple seasonality are now possible. The derivation procedure can also be extended and generalized to the additive case. The simple structure of the proposed model retains the data efficiency and ease of implementation of the original ES family, while leveraging cross-learning to shape the seasonality over related time series. Moreover design simplicity allows inspection of the learnt parameter and understanding of the model outcome, possibly answering to a lack of explanaibility of ML models. There are of course limitations to the approach. Requiring time-dependent co-variates, the model is limited in those scenarios - very uncommon in practice, but not impossible - where no calendar information are provided. Also, in the case of asymmetric data, the cross-learning process, smoothing the extrema, could impact the forecasting at the distribution's tail. As it is, PES can not be used to forecast a new time series for which no data is available. A possible workaround would be to assign to the new series a value of α and θ averaged over those of related time series. The natural continuation of the presented work is the introduction of the trend component, either as an operative block within the cell or as part of the context vector. Another interesting enhancement would be the promotion of the α^i parameters from scalar to vector. The latter could be beneficial to further capture dependencies between successive time steps and variations over the time series period. Structure's simplicity however does not impact performances, we have shown how PES is able to favorably compare to two more complex solutions, namely DeepAR and DeepSSMs, both coming from the Amazon Research Labs and commercially available.

REFERENCES

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

Vassilis Assimakopoulos and K. Nikolopoulos. The theta model: A decomposition approach to forecasting. *International Journal of Forecasting*, 16:521–530, 10 2000. doi: 10.1016/S0169-2070(00)00066-2.

Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing, 2012. URL `https://arxiv.org/abs/1209.5145`.

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

Jan G. De Gooijer and Rob J. Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.

James Durbin and Siem J. Koopman. *Time Series Analysis by State Space Methods*. Number 019-961199 in Oxford Statiscal Science Series. Oxford University press, 2001.

Clive WJ Granger and Roselyne Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of time series analysis*, 1(1):15–29, 1980.

Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990.

Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 1957.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts: Melbourne, Australia., 2021. URL https://otexts.com/fpp3/index.html. Accessed on 2022-03-09.

Rob J. Hyndman and Baki Billah. Unmasking the theta method. *International Journal of Forecasting*, 19(2):287–290, 2003.

Rob J. Hyndman, Anne B. Koehler, Ralph D. Snyder, and Simone Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting*, 18(3):439–454, 2002.

Rob J. Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008a.

Rob J. Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. Forecasting with exponential smoothing: the state space approach. https://robjhyndman.com/expsmooth/, 2008b. Website with supplementary materials for the homonymous book [Online].

Rob J. Hyndman et al. The interaction between trend and seasonality. *International Journal of Forecasting*, 20(4):561–563, 2004.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 2022a. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL https://www.sciencedirect.com/science/article/pii/S0169207021001874.

Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L. Winkler. The m5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, 2022b. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.10.009. URL https://www.sciencedirect.com/science/article/pii/S0169207021001722.

McKinsey & Company. Global survey - the state of ai in 2020. https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/global-survey-the-state-of-ai-in-2020, 2020. McKinsey Analytics [Online].

C Carl Pegels. Exponential forecasting: Some new variations. *Management Science*, pp. 311–315, 1969.

Christopher Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux.jl - A julia library for neural differential equations. *CoRR*, abs/1902.02376, 2019. URL https://arxiv.org/abs/1902.02376.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31:7785–7794, 2018.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

Matthias W Seeger, David Salinas, and Valentin Flunkert. Bayesian intermittent demand forecasting for large inventories. *Advances in Neural Information Processing Systems*, 29, 2016.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186, 2020.

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

Ralph D. Snyder, J. Keith Ord, and Adrian Beaumont. Forecasting the intermittent demand for slow-moving inventories: A modelling approach. *International Journal of Forecasting*, 28(2):485–496, 2012. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2011.03.009. URL https://www.sciencedirect.com/science/article/pii/S0169207011000781.

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

George Udny Yule. On a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

A APPENDIX

A.1 VISUALIZATION

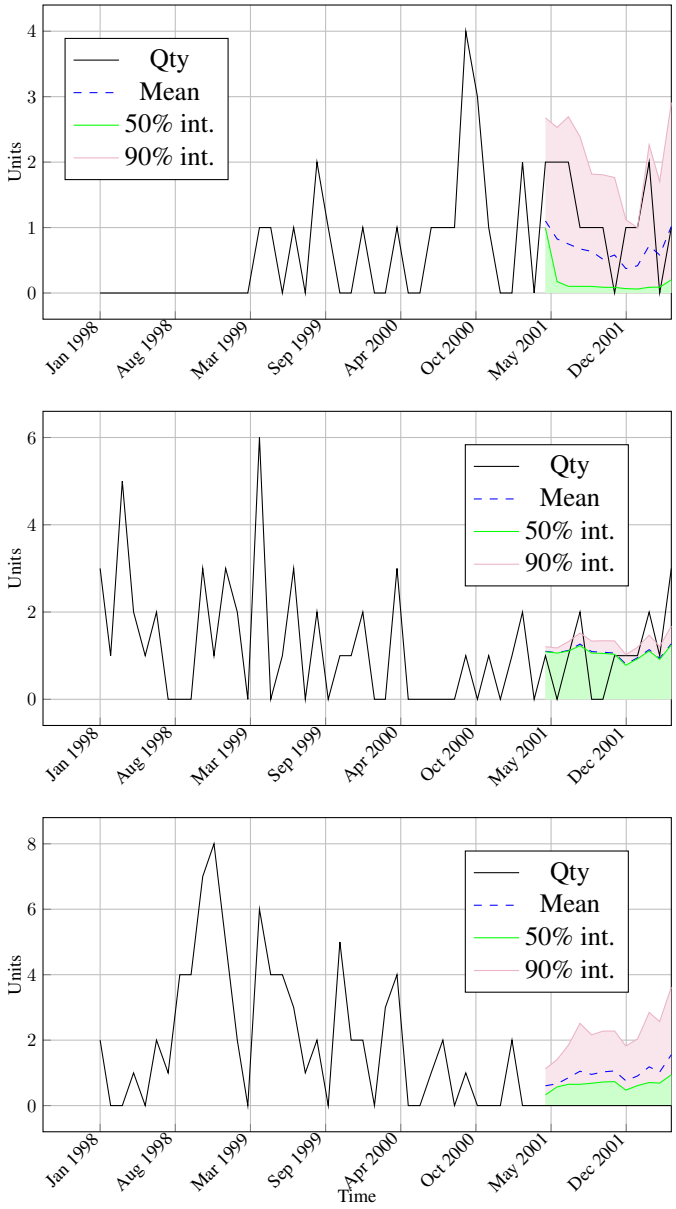

Figure 5: Example of forecast over parts dataset. From top to bottom id 495, id 842 and id 991.

A.2 ENCODING PARAMETERS

Parameters α^i and seasonal factors $\mathbf{s}^g_{1:m_g}$ are subject to constraints. This appendix introduce these constraints and how they had been implemented in our solution.

Smoothing factors Historically the smoothing factor α - and in general any smoothing factor in the ES framework, i.e. β and γ - is expected to be in the range $[0, 1]$: $\alpha = 0$ means that the method relies completely on the last *smoothed* value; conversely for $\alpha = 1$ the method focuses its attention over the last *seen* value. In Hyndman & Athanasopoulos (2021) different boundaries are reported

for β and γ, specifically: $0 \leq \beta^* \leq 1$, $\beta = \alpha\beta^*$; $0 \leq \gamma \leq (1-\alpha)$. These boundaries ease the computation of the exponential smoothing methods in SSM form.

We anyway kept the historical boundaries. In the PES model proposed we had only the α parameter - the only one with untouched domain in the Hyndman et al. revision -, no trend and several seasonal profiles which did not require a smoothing procedure. To keep the α^i restricted, we used proxies $\tilde{\alpha}^i$ which are then scaled via a bounded sigmoid. We set *l*ower *b*ound $lb = 0.005$ and an *u*pper *b*ound $ub = 0.95$, such that

$$\alpha^i = (ub - lb)\frac{1}{1 + \exp(-\tilde{\alpha}^i)} + lb = \frac{0.9}{1 + \exp(-\tilde{\alpha}^i)} + 0.05$$

Seasonal profiles Two types of seasonality are the most used in practice: additive and multiplicative, with the option to treat multiplicative case as additive taking a logarithm transformation of the observations. In both cases we expect the seasonality to be stationary and repeat itself from period to period. Abrupt changes, especially for additive seasonality, will most likely be incorporated either into the trend component, the error or both. To ensure that the seasonality is stationary we have to check that

$$\sum_{t=1}^{m_g} s_t^g = \begin{cases} 0 & \text{if additive} \\ m_g & \text{if multiplicative} \end{cases}$$

To control the magnitude of the patterns we employed proxies $\tilde{\mathbf{s}}^g_{1:m_g}$, as we did for the scaling factor, and two separate scaling schema:

$$\begin{aligned} \hat{s}_t^g &= \frac{exp(\tilde{s}_t^g)}{\sum_{t=1}^{m_g} exp(\tilde{s}_t^g)} t, \\ \bar{s}_t^g &= \frac{\sum_{t=1}^{m_g} \hat{s}_t^g}{m_g}, \\ s_t^g &= \hat{s}_t^g - \bar{s}_t^g, \end{aligned}$$

for an additive seasonality;

$$s_t^g = \frac{exp(\tilde{s}_t^g)}{\sum_{t=1}^{m_g} exp(\tilde{s}_t^g)} * m_g$$

for a multiplicative one.

A.3 Scaling operators

Imagine that the data expose a *W*eek-*o*f-*Y*ear (woy) and *moy* seasonality, both multiplicative. Two scaling vectors will then be defined and shared, i.e. $\mathbf{s}^{woy}_{1:52} = [s^{woy}_1, \dots, s^{woy}_{52}]$ and $\mathbf{s}^{moy}_{1:12} = [s^{moy}_1, \dots, s^{moy}_{12}]$. For the i-th time series at time t, for $t^g = t \ mod(m_g)$, the application of the scaling operator $s(\cdot)$ will be

$$s(z_t^i, \mathbf{s}^{woy}, \mathbf{s}^{moy}, \mathbf{x}_t^i) = \frac{z_t^i}{s^{woy}_{t^{woy}} * s^{moy}_{t^{moy}}}$$

The inverse operator will therefore have the following form

$$s^{-1}(z_t^i, \mathbf{s}^{woy}, \mathbf{s}^{moy}, \mathbf{x}_t^i) = z_t^i * (s^{woy}_{t^{woy}} * s^{moy}_{t^{moy}})$$

If one of the two seasonality is additive, for example the weekly one, we will proceed in a descending order, removing the seasonal effects at higher frequency before those at the immediate lower level. In our example we begin removing the moy effect before treating the woy

$$s(z_t^i, \mathbf{s}^{woy}, \mathbf{s}^{moy}, \mathbf{x}_t^i) = \frac{z_t^i}{s^{moy}_{t^{moy}}} - s^{woy}_{t^{woy}}$$

The inverse operator will progress in ascending order, applying back the seasonality at lower frequency before those at a higher level. That is

$$s^{-1}(z_t^i, \mathbf{s}^{woy}, \mathbf{s}^{moy}, \mathbf{x}_t^i) = (z_t^i + s^{woy}_{t^{woy}}) * s^{moy}_{t^{moy}}$$