The supply chain of Robert Bosch Power Tools is characterized by short lead-times to its customers (48hours) but long replenishment, production and supplier lead times (up to 18 months). Therefore a precise demand forecast is crucial. This is currently done in a standard demand managment software which creates point forecasts of all forecast relevant base combinations up to the next 24 months. These are then fed into a inventory management system which calculates the optimal stock level and derives and automatically triggers then replenishment decisions using a complex reorder point heuristic. The product *Scenario-Based Total Demand Forecasting* of the PT-LOG program *LISA* calculates distribution forecasts of the total demand of all forecast relevant base combinations up to the next 18 months. Certain percentiles of these distributions are then loaded into the standard demand planning system of PT. These demand distribution forecasts can then be used downstream to more precisely optimize replenishment and inventory decisions.

1 Introduction

[X longer part about the setup and circumstancence of the LISA project and PT planning X]

This report is structured as follows. We begin by reviewing related work in Section 2 before we introduce our approach in Section 3. Then we introduce how the model perfromance is evaluated in Section 4. Afterwards we describe our implementation in Section 5 and show our results in Section 6. Next, we describe possible future steps in Section 7. We finish with a conclusion in Section 8.

2 Related Work

3 Model

The entities of multi-variate time series will be denoted $z_t^i \in \mathbb{R}_{\geq 0}$ for series $i \in \mathfrak{I}$ and time $t \in [1, T]$.

The series set $\mathfrak{I}$ consists of all forecast relevant base items $\mathfrak{I} = X_{b \in \mathfrak{B}} \mathfrak{I}_b = \times_{b \in \mathfrak{B}} \mathfrak{A}_b \times \mathfrak{W} \times \mathfrak{R} \times \mathfrak{C}$ where $\times$ denotes the cross product of sets, $\mathfrak{B}$ is the set of Business Units, $\mathfrak{I}_b$ is the set of forecast relevant base types for Business Unit b, $\mathfrak{A}_b$ denotes the set of articles belonging to Business Unit b, $\mathfrak{W}$ denotes the set of warehouses, $\mathfrak{R}$ denotes the set of regions like a regional sales organisation RSO or a country sales organisation CSO, and $\mathfrak{C}$ denotes the set of customer groups.

The time interval $[1, T]$ can be split into a conditioning range $[1, t_0 - 1]$ and a prediction range $[t_0, T]$. In the conditioning range the time series values are known, in the prediction range the time series values are unkown or at least to be predicted. We make sure that each time series $i \in \mathfrak{I}$ has data within the full conditioning range $[1, t_0 - 1]$ by filling up shorter time series by leading Zeros.

In addition to the time series values itself z_t^i we also have m-dimensional covariates $\mathbf{c}_t^i \in \mathbb{R}^m$ for series $i \in \mathfrak{I}$ and time $t \in [1, T]$. It is assumed that the covariate values are both known for the conditioning range and a prediction range.

The goal is to predict the conditional probabilities

$$P\left(z_{[t_0,T]}^i \middle| z_{[1,t_0-1]}^i, \mathbf{c}_{[1,T]}^i\right) \quad \forall i \in \mathfrak{I} \tag{1}$$

We assume that at each time step the values are drawn in an independent and identically distributed way. Then Eq. (10) can be expanded as an autoregressive product

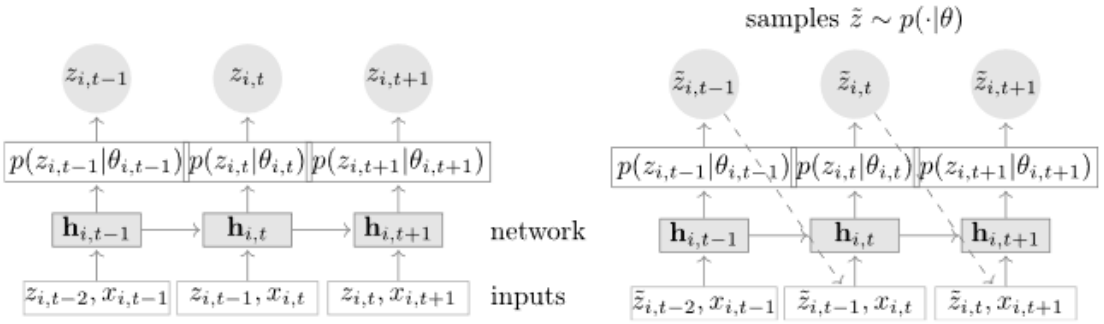


Fig. 1: Figure 3 from Ref. [1]. Summary of the model: Training is shown on the left, prediction is shown on the right. At each time step t, the inputs to the network are the covariates $\mathbf{c}_t^i$, the target value at the previous time step z_{t-1}^i, and the previous network output $\mathbf{h}_{t-1}^i$. The network output $\mathbf{h}_t^i = \mathfrak{h}\left(\mathbf{h}_{t-1}^i, z_{t-1}^i, \mathbf{c}_t^i, \Theta\right)$ is then used to compute the parameters $\theta_t^i = \theta(\mathbf{h}_t^i, \Theta)$ of the likelihood $p(z|\theta)$, which is used for training the model parameters. For prediction, the history of the time series z_t^i is fed in for $t < t_0$, then in the prediction range (right) for $t \geq t_0$ a sample $z_t^i \sim p(.|\theta_t^i)$ is drawn and fed back for the next point until the end of the prediction range, generating one sample trace. Repeating this prediction process yields many traces that represent the joint predicted distribution.

$$P\left(z_{[t_0,T]}^i \middle| z_{[1,t_0-1]}^i, \mathbf{c}_{[1,T]}^i\right) = \prod_{t=t_0}^{T} P\left(z_t^i \middle| z_{[1,t-1]}^i, \mathbf{c}_{[1,T]}^i\right) \quad \forall i \in \mathfrak{I} \tag{2}$$

We are currently working on two approaches which are introduced now.

3.1 Approach 1: DeepAR: Probabilistic forecasting with autoregressive recurrent networks

For the first approach we orientate us at Ref. [1] and [2] and the model is summarized in Fig. 1. The model is based on the ideas of Ref. [3] and [4].

3.1.1 Model

Starting at Eq. (2), we further assume that the conditional probabilities at each time step can be modeled as likelihood factors

$$P\left(z_t^i \middle| z_{[1,t-1]}^i, \mathbf{c}_{[1,T]}^i\right) = p\left(z_t^i \middle| \theta(\mathbf{h}_t^i, \Theta)\right) \tag{3}$$

where θ models a function which outputs all parameters of the likelihood distribution. It is parametrized by the output of an **autoregressive recurrent network**

$$\mathbf{h}_t^i = \mathfrak{h}\left(\mathbf{h}_{t-1}^i, z_{t-1}^i, \mathbf{c}_t^i, \Theta\right) \tag{4}$$

where $\mathfrak{h}$ is a function which is implemented by a **multi-layer recurrent neural network with LSTM cells** parametrized by Θ.

The likelihood $p\left(z_t^i \middle| \theta(\mathbf{h}_t^i, \Theta)\right)$ (as a function of θ for a fixed z_t^i) is a fixed distribution with parameters that come out of the function $\theta(\mathbf{h}_t^i, \Theta)$.

In the original sequence-to-sequence setup of Ref. [3] and [4], there is a encoder network learned on the training range and a decoder network learned on the prediction range. In our context based on Ref. [1], these two networks are identical in architecture and parameters Θ. The initial states of the encoder $\mathbf{h}_0^i$ and z_0^i are initialized to zero. The initial hidden state of the decoder $\mathbf{h}_{t_0-1}^i$ is obtained by computing Eq. (4) for $t \in [1, t_0 - 1]$, where all required quantities are observed.

With trained network parameters Θ, we can than draw joint samples $\tilde{z}_{[t_0,T]}^i \sim P\left(z_{[t_0,T]}^i \middle| z_{[1,t_0-1]}^i, \mathbf{c}_{[1,T]}^i\right)$ by ancestral sampling. By drawing many of these joint samples we can calculate the quantiles of interest.

3.1.2 Likelihood function

The likelihood $p(z|\theta)$ defines the "noise model" and it should match the data as close as possible. It models the probability distribution of the time series value in the next time step. All parameters of the likelihood are a direct outcome out of the neural network.

Many likelihood models are possible in this context as long as samples from the distribution can be obtained cheaply, and the log-likelihoodand its gradients with respect to the parameters can be evaluated.

In our case of sales quantities $z_t^i \in \mathbb{R}_{\geq 0}$, the negative-binomial likelihood is a good choice, see e.g. Ref. [5] or [6].

We parameterize the negative binomial distribution by its mean $\mu \in \mathbb{R}_{\geq 0}$ and its shape parameter $\alpha \in \mathbb{R}_{\geq 0}$,

$$p_{NB}(z|\mu,\alpha) = \frac{\Gamma(z+\frac{1}{\alpha})}{\Gamma(z+1)\Gamma(\frac{1}{\alpha})}\left(\frac{1}{1+\alpha\mu}\right)^{\frac{1}{\alpha}}\left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z} \tag{5}$$

with

$$\mu = \mu(\mathbf{h}_t^i) = \log\left(1+\exp\left(\mathbf{w}_\mu^T\mathbf{h}_t^i + b_\mu\right)\right) \tag{6a}$$

$$\alpha = \alpha(\mathbf{h}_t^i) = \log\left(1+\exp\left(\mathbf{w}_\alpha^T\mathbf{h}_t^i + b_\alpha\right)\right) \tag{6b}$$

where both parameters are obtained from the network output by a fully-connected layer with softplus activations as to ensure positivity. In this parameterization of the negative binomial distribution, the shape parameter α scales the variance relative to the mean, i.e. $Var[z] = \mu + \mu^2\alpha$.

3.1.3 Training

For the training, we define the loglikelihood function as

$$\mathcal{L} = \sum_{i\in\mathfrak{I}^{\mathrm{Tr}}}\sum_{t=t_0}^{T}\log(p(z_t^i|\theta(\mathbf{h}_t^i,\Theta))) \tag{7}$$

with $\mathfrak{I}^{\mathrm{Tr}}$ as the training set where z_t^i and $\mathbf{c}_t^i$ is known for all $t \in [1,T]$ and $i \in \mathfrak{I}^{\mathrm{Tr}}$. The loglikelihood function is now maximized with respect to Θ, all parameters of $\mathfrak{h}(.)$ and $\theta(.)$. The optimization can be done directly via stochastic gradient descent by computing gradients with respect to Θ. As here, encoder and decoder is identical, we can set t_0 to Zero in Eq. (7) such that we optimize over the condition and prediction range. For each timeseries in the dataset $\mathfrak{I}$, we generate multiple training instances by selecting from the original timeseries windows with different starting points but total length T and the relative lengths of the conditioning and prediction ranges fixed for all training examples. We allow also start points before the beginning of a given timeseries and fill up the missing points with Zeros. By this, the model can learn also pattern for new items.

3.1.4 Scale handling

Our time series despict a power law, so we have many tiny time series and a few very big ones. Out of this two problems arise, see Ref. [1].

Firstly, the autoregressive nature of the model means that the input and output scales directly with the time series values but the non-linearities of the network in between have only a limited operating range. Therefore the neural net would need to learn additional mappings which is hard. Ref. [1] addresses this issue by dividing the autoregressive inputs z^i by an item-dependent scale factor ν^i, and conversely multiplying the scale-dependent likelihood parameters by the same factor. The negative binomial likelihood parameters of Eq. (6) become

$$\mu = \nu^i \log\left(1 + \exp\left(\mathbf{w}_\mu^T \mathbf{h}_t^i + b_\mu\right)\right) \tag{8a}$$

$$\alpha = \frac{\log\left(1 + \exp\left(\mathbf{w}_\alpha^T \mathbf{h}_t^i + b_\alpha\right)\right)}{\sqrt{\nu^i}} \tag{8b}$$

A practical solution in Ref. [1] for the item-dependent scale factor ν^i is to use the time series average

$$\nu^i = 1 + \frac{1}{t_0} \sum_{t=1}^{t_0} z_t^i \tag{9}$$

The second problem is that the imbalance in the data means that a stochastic optimization procedure that picks training instances uniformly at random will visit the small number time series with large scales very infrequently, resulting in those time series being underfitted. In our demand forecasting setting often these large scale time series show different behaviour. This is than not learned enough by the model, exactly there were it is most important as these time series represent the highest business value. Ref. [1] solves this by using a weighted sampling scheme that sets the probability of selecting a window from an example with scale ν^i proportional to ν^i.

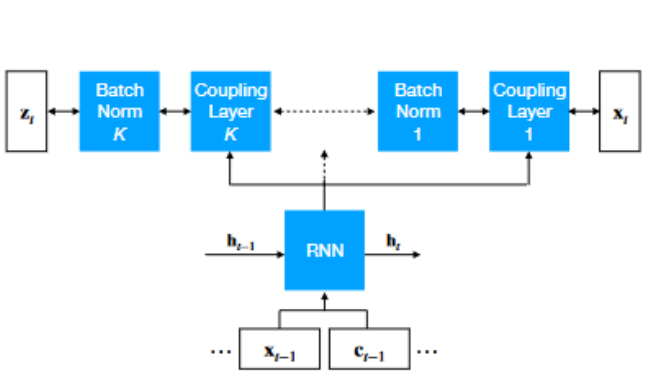
3.2 Approach 2: Multi-variate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows

For the second approach we orientate us at Ref. [7].

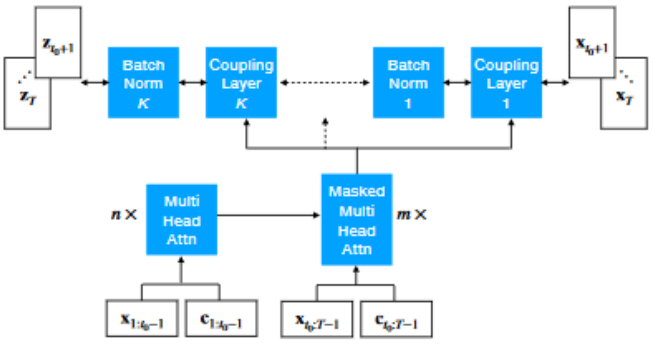
3.2.1 Differences between the two Approaches

The main differences to the first approach (based on Ref. [1]) are

- In the second approach all time series are modelled joinly as one complex multi-variate probability distribution (each time series is one addtional dimension) while in the first approach there is only one scalar probability distribution which needs to fit for all time series. This enables the model to take effects like cannibalization and other cross-time series effects into account.
- In the second approach using conditional normalized flows (see Ref. [8] and [9]) the distribution can be arbitrarily complex while in the first approach one has to choose one distribution class for all time series, e.g. nonnegative binomial for count data.
- Instead of an RNN with LSTM cells in approach 1, here one can use either a RNN with GRUs (see Ref. [10]) or a Transformer model using Self-Attention (see Ref. [11])
- For categorical features of the covariates approach 2 uses embeddings (see Ref. [12])



(a) Fig 1 of Ref. [7]: RNN Conditioned Real NVP model schematic at time t, consisting of K blocks of coupling layers and Batch Normalization, where in each coupling layer we condition the time series (in the pic denoted x_t) and its transformations on the state of a shared RNN from the previous time step and its covariates which are typically timedependent and time independent features.



(b) Fig 2 of Ref. [7]: Transformer Conditioned Real NVP model schematic consisting of an encoder-decoder stack where the encoder takes in some context length of time series and then uses it to generate conditioning for the prediction length portion of the time series via a causally masked decoder stack. The output of the decoder is used as conditioning to train the flow. Note that the positional encodings are part of the covariates and unlike the RNN model,here all time series time points are trained in parallel.

Fig. 2: The two main different model architectures of approach 2 based on Ref. [7]

3.2.2 Model

The goal is to predict the conditional probability

$$P\left(\mathbf{z}_{[t_0,T]}\middle|\mathbf{z}_{[1,t_0-1]},\mathbf{c}_{[1,T]}\right) \tag{10}$$

where $\mathbf{z}=[z^i]_{i\in\mathfrak{I}}$ is the multi-variate time series vector and $\mathbf{c}=\{\mathbf{c}^i\}_{i\in\mathfrak{I}}$ are the covariates over all times series. Fig. 2 shows schematically the 2 possible architectures within the second approach.

3.2.3 Conditioned Real NVP

Based on Ref. [8] and [9] the 2nd approach uses conditioned real non-volume preserving flows (Real NVP) in order to model complex multi-dimensional density distributions. Fig. 3 describes the main idea.

4 Performance Evalution

In this section, we define how we evaluate the performance of our solution.

4.1 Forecast Horizon

In the first phase, we forecast 6 months ahead. All horizons will be weighted equally in the performance evaluation:

$$w_t^T=\frac{1}{T-t_0}\quad\forall t\in[t_0,T]. \tag{11}$$

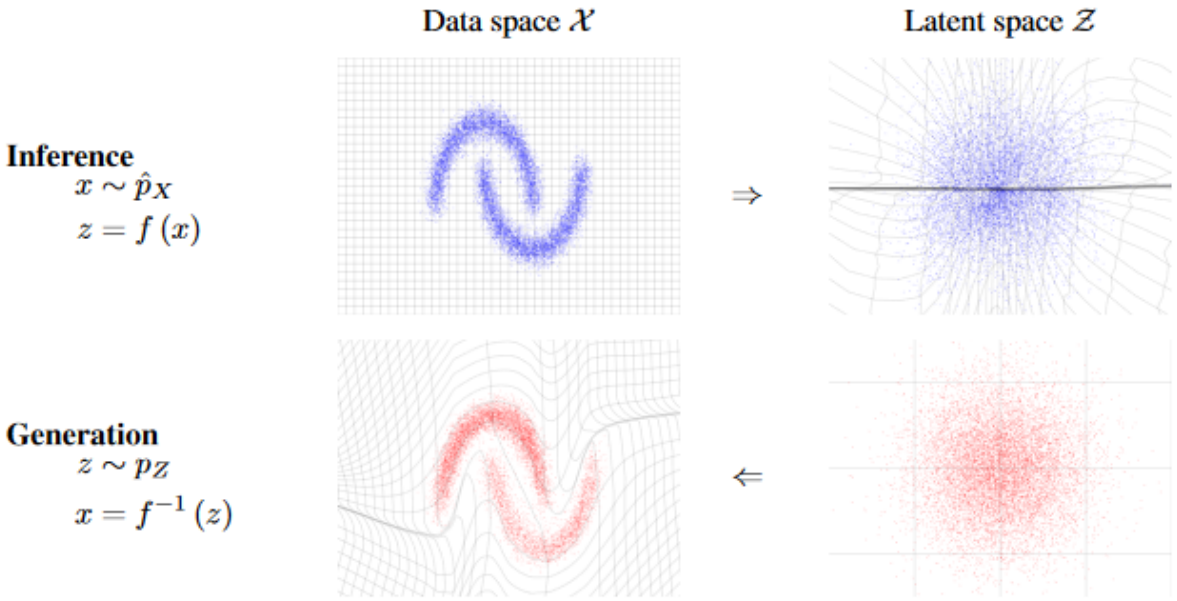


Fig. 3: Fig. 1 from Ref. [9]. Real NVP learns an invertible, stable, mapping between a data distribution and a latent distribution (typically a Gaussian).

4.2 Historic Horizon

We will use the last 24 months to condition our model. In addition we will use this data also to scale our errors using the following weights

$$w_t^P = \frac{\gamma^{t_0-1-t}}{\sum_{r=2}^{t_0-1}\gamma^{t_0-1-r}} \quad \forall t \in [2, t_0-1]. \tag{12}$$

$\gamma \in [0,1]$ is a shrinkage factor which reduces the weights for times farer in the past. Here we will use $\gamma = 0.95$.

4.3 Point Forecast

Even when our goal is to forecast the demand density, we will measure a forecast quality on a point forecast to compare the new solution with our existing benchmark solutions. We use the median to map the density to a point:

$$\hat{z}_t^i = \hat{z}_t^i(0.5) = Median\left(P\left(z_t^i \middle| z_{[1,t-1]}^i, \mathbf{c}_{[1,T]}^i\right)\right) \tag{13}$$

4.3.1 Root Mean Squared Scaled Error

Then for the forecast quality measure, we use the Root Mean Squared Scaled Error (**RMSSE**) as in REf. [13]:

$$\mathbf{RMSSE}^i = \sqrt{\frac{\sum_{t=t_0}^{T} w_t^T \left(z_t^i - \hat{z}_t^i\right)^2}{\sum_{t=2}^{t_0-1} w_t^P \left(z_t^i - z_{t-1}^i\right)^2}} \tag{14}$$

where z_t^i is the actual value of the time series i at time t, w_t^T is the horizon weight from Eq. (11) and w_t^P is the past weight from Eq. (12).

The nominator of Eq. (14) can be also interpreted as the expected weighted average forecast error using the naive forecasting method $\hat{z}_t^i = z_{t-1}^i$. Therefore, **RMSSE** can interpreted as the ratio of the weighted average forecasting error over the prediction range of our forecast vs the expected (based on the past) weighted average forecast error using the naive forecasting.

For the forecast quality measure of a set of forecasts we use the **Weighted RMSSE(WRMSSE)**,

$$\mathbf{WRMSSE} = \sum_{i \in \mathfrak{I}} w_i \mathbf{RMSSE}^i \tag{15}$$

where w_i are the weights for each series defined by

$$w_i = \frac{\mathfrak{p}^i * \left(\sum_{t=1}^{t_0-1} w_t^P z_t^i\right)}{\sum_{j \in \mathfrak{I}} \mathfrak{p}^j * \left(\sum_{t=1}^{t_0-1} w_t^P z_t^j\right)} \tag{16}$$

where $\mathfrak{p}^i$ is the production price over the time $t \in [1, t_0 - 1]$ of time series i and w_t^P is the past weight from Eq. (12).

4.3.2 Forecast Accuracy and FC Bias

At Bosch Power Tools we have official business KPIs for forecast quality measurement: Forecast Accuracy and FC Bias.

First, we define the absolute relative error as

$$\mathbf{ARE}_t^i = \begin{cases} \frac{|z_t^i - \hat{z}_t^i|}{z_t^i} & \text{if } z_t^i > 0 \\ 0 & \text{if } z_t^i = \hat{z}_t^i = 0 \\ \infty & \text{if } z_t^i = 0 \wedge \hat{z}_t^i > 0 \end{cases} \tag{17}$$

In order to cope with the high outliers and the values of infinity we define an truncated absolute relative error

$$\mathbf{TARE}_t^i = min(\mathbf{ARE}_t^i, \rho^{\mathbf{TARE}}) \tag{18}$$

where here the maximal value of **TARE** $\rho^{\mathbf{TARE}} = 130\%$.

The Forecast Accuracy is now defined as

$$\mathbf{FCA}_t = 100 * max\left(1 - \sum_{i \in \mathfrak{I}} \omega_t^i \mathbf{TARE}_t^i, 0\right) \tag{19}$$

with forecast quality weights

$$\omega_t^i = \frac{\mathfrak{p}^i \frac{z_t^i + \hat{z}_t^i}{2}}{\sum_{j \in \mathfrak{I}} \mathfrak{p}^j \frac{z_t^j + \hat{z}_t^j}{2}} \tag{20}$$

At Bosch Power Tools we have an average lead time of 7 weeks therefore we use horizon 2

$$\mathbf{FCA} = \mathbf{FCA}_{t_0+2} \tag{21}$$

Forecast Bias is calculated in a very similar way.

First, we define the relative error as

$$\mathbf{RE}_t^i = \begin{cases} \frac{z_t^i - \hat{z}_t^i}{z_t^i} & \text{if } z_t^i > 0 \\ 0 & \text{if } z_t^i = \hat{z}_t^i = 0 \\ \infty & \text{if } z_t^i = 0 \wedge \hat{z}_t^i > 0 \end{cases} \tag{22}$$

In order to cope with the high outliers and the values of infinity we define an truncated relative error

$$\mathbf{TRE}_t^i = min(\mathbf{RE}_t^i, \rho^{\mathbf{TRE}}) \tag{23}$$

where here the maximal value of **TRE** $\rho^{\mathbf{TRE}} = 130\%$.

The Forecast Bias is now defined as

$$\mathbf{FCB}_t = 100 * \sum_{i \in \mathfrak{I}} \omega_t^i \mathbf{TRE}_t^i \tag{24}$$

with forecast quality weights as in Eq. (20).

At Bosch Power Tools we have an average lead time of 7 weeks therefore we use horizon 2

$$\mathbf{FCB} = \mathbf{FCB}_{t_0+2} \tag{25}$$

4.3.3 Overall Weighted Error

In order to have only one score to judge if one model is better than the other we define an Overall Weighted Error in a similar way as in Ref. [14].

For this we first normalize our single measures by

$$\mathbf{normWRMSSE} = \frac{\mathbf{WRMSSE}(model)}{\mathbf{WRMSSE}(naive)} \tag{26}$$

and

$$\mathbf{normFCE} = \frac{1 - \mathbf{FCA}(model)/100}{1 - \mathbf{FCA}(naive)/100} \tag{27}$$

where $\mathbf{WRMSSE}(model)$ and $\mathbf{FCA}(model)$ are the measures defined in Eq. (15) and (25) using the models forecast. While $\mathbf{WRMSSE}(naive)$ and $\mathbf{FCA}(naive)$ are the measures defined in Eq. (15) and (25) using the naive forecast

$$\text{naive forecast} = \hat{z}_t^i = z_{t_0-1}^i \quad \forall t \in [t_0, T] \tag{28}$$

Now the Overall Weighted Error is defined by

$$\mathbf{OWE} = \frac{\mathbf{normWRMSSE} + \mathbf{normFCE}}{2} \tag{29}$$

4.4 Probabilistic Forecast

For the probabilistic forecast we first define the Pinball loss (**PL**) for a specific quantile u as

$$\mathbf{PL}_t^i(u) = (z_t^i - \hat{z}_t^i(u))u\mathbb{1}\{z_t^i \leq \hat{z}_t^i(u)\} + (\hat{z}_t^i(u) - z_t^i)(1-u)\mathbb{1}\{z_t^i > \hat{z}_t^i(u)\} \tag{30}$$

where $\mathbb{1}$ is the indicator function (it is 1 if the condition is true otherwise it is 0), and $\hat{z}_t^i(u)$ is the u-quantile of the generated probabilistic forecast Eq. (10).

The interpretation of the pinball loss is that the ratio of one unit overforecast vs one unit underforecast is equal to the odds of quantile u. If one optimizes then in respect to this loss than probability of forecast beeing smaller or equal quantile prediction is equal to u which is then exactly how a quantile is defined.

Now we calculate the scaled pinball loss (**SPL**) for a specific time series i

$$\mathbf{SPL}^i(u) = \sqrt{\frac{\sum_{t=t_0}^{T} w_t^T \mathbf{PL}_t^i(u)}{\sum_{t=2}^{t_0-1} w_t^P \left|z_t^i - z_{t-1}^i\right|}} \tag{31}$$

where w_t^T is the horizon weight from Eq. (11) and w_t^P is the past weight from Eq. (12) .

To optimize not only one quantile but the whole distribution we minimize the sum of the scaled pinball loss of Eq. (31) over many quantiles

$$\mathbf{SPL}^i = \sum_{u \in \mathfrak{U}} w_u^{\mathfrak{U}} \mathbf{SPL}^i(u) \tag{32}$$

with

$$\mathfrak{U} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.97, 0.99\} \tag{33}$$

with assoziated neutral weights

$$w_u^{\mathfrak{U}} = \frac{1}{\#\mathfrak{U}} \quad \forall u \in \mathfrak{U}. \tag{34}$$

For the forecast quality measure of a set of probabilistic forecasts we use then the **Weighted SPL(WSPL)**,

$$\mathbf{WSPL} = \sum_{i \in \mathfrak{I}} w_i \mathbf{SPL}^i \tag{35}$$

with the same weights w_i as from Eq. (16).

5 Implementation

In this section, we describe how the model has been implemented.

6 Results

In this section, we describe how the model has been implemented.

7 Future Work

8 Conclusion

References

[1] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181 – 1191, 2020.

[2] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. Gluonts: Probabilistic time series models in python. *CoRR*, abs/1906.05264, 2019.

[3] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[5] Nicolas Chapados. Effective bayesian modeling of groups of related count time series. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1395–II–1403. JMLR.org, 2014.

[6] 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.

[7] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows, 2020.

[8] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation, 2018.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[10] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables, 2016.

[13] University of Nicosia. The m5 competition: Competitors' guide, 2020.

[14] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54 – 74, 2020. M4 Competition.