Abstract—Short-term load forecasting is typically used by electricity market participants to optimize their trading decisions and by system operators to ensure reliable grid operation. In particular, it allows the latter to foresee potential power imbalances and other critical grid states and thereafter, to enforce appropriate mitigation actions. Especially, forecasting critical grid states such as congestions, plays an essential role in this context. This paper proposes a recurrent neural network that is trained to forecast day-ahead time-series and prediction intervals for residual loads. Moreover, a comprehensive overview on probabilistic evaluation metrics is given. The ignorance score and the quantile score are used during the training whereas other metrics are for evaluation as they facilitate comparability between the different forecasting approaches with the naïve baselines. The proposed deep learning model can be both specified as a parametric or as a non-parametric model and delivers reliable forecasts for day-ahead purposes.

Index Terms—Probabilistic load forecasting, time-series, machine learning, RNN, quantile regression, energy flexibility, congestion management

I. INTRODUCTION

Today’s power grids are challenged by an ever increasing electricity demand, especially in growing cities. This effect particularly impacts the distribution grid, but it can also cause congestion at the interface between transmission and distribution systems.

At the same time an increase in the number of distributed energy resources (DERs) is observed in most distribution systems. As a result, the distribution system operator (DSO) needs to identify the energy generated in its own system and how much remaining capacity (residual load) needs to be supplied through the primary substations, i.e. via the transmission system. It is integral to forecast when and for how long congestions of this type might occur in order to initiate counteractions on the transmission or distribution side. This is particularly important during high-load periods, e.g. winter, when the grid is often operated under lower reserve margins at the cost of reliability. On the distribution side, modern grids contain flexible assets that can be reserved for congestion mitigation and dispatched upon request. These flexibilities can be requested at short notice on the electricity market. However, to avoid unnecessary flexibility acquisition, accurate insights into the near-future grid state is vital. For this reason, the proposed work accommodates a model that computes day-ahead time-series forecasts including prediction intervals.

While monitoring and forecasting the grid state is well-established in the transmission system, DSOs often lack monitoring and suitable grid control. Thus, the contribution of this work is the design of a machine learning (ML)-based, probabilistic forecasting model that only requires a small amount of measurements, which are typically already available in the operators’ Supervisory Control And Data Acquisition (SCADA) system (such as historic measurements at substation level) to indicate critical states, their time of occurrence and duration.

II. PROBABILISTIC LOAD FORECASTING

Major uncertainties associated with vertical grid load forecasting are primarily caused by the fact that residential energy consumption patterns can be highly variable; the energy consumption often depends non-linearly on weather conditions [1], and unusual behavior during special calendar days and changing socio-economic circumstances [2]. Since we are interested in the residual load, local power production plays also a key role. Volatile DER generation depends strongly on the weather conditions, such that the residual load is subject to this additional uncertainty.

A. Related Work

Probabilistic methods attempt to provide information on the uncertainty of the discrete random variable’s expected value based on conditional kernel density estimation as presented in [3] where a mixture density network for a Gaussian model is applied. The authors obtain forecasts for domestic load profiles with granularity below an hour. In terms of the granularity and the size of the loads, [4] presents a model that accommodates nearly all aspects necessary to generate predictions for the later presented case study. The authors develop a Gaussian process model, but lack to benchmark their results with comparable models. In [5] a hierarchical approach is proposed, applying quantile regression (QR) and determining joint distributions of the random variables for a long-term load forecast.
The work conducted in [6] compares five load forecasting models, including the uncertainty due to meteorological forecasts. However, it predicts over a monthly prediction horizon.

In [7], the authors compare bidirectional long-short-term memory units (LSTM) based on QR and Gaussian parameters for probabilistic forecasting of aggregated PV and wind generation, load and electricity prices. In a second step, a copula-based sampling method is applied to generate scenarios for a scenario-based stochastic optimization of day-ahead electricity market bids.

Similarly, parametric and non-parametric ML-based forecasts are developed in this work in the context of day-ahead forecasting of the residual electric load at highly aggregated grid nodes. The focus is on the comparability of the different approaches. For this purpose, relevant probabilistic metrics are reviewed and utilized to facilitate a comprehensive comparison between the selected models and the baselines. These simple baseline models are extended to the probabilistic setting to increase the interpretability of the results.

B. Probabilistic Metrics

Predictive distributions of the random variable can either be obtained through non-parametric, semi-parametric or parametric methodologies. Non-parametric ones do not make any assumptions on the shape of the probability distribution (PDF) of the random variable. Instead, they focus on determining regression coefficients, e.g. a set of quantile samples applying the least absolute error estimator as described in [8]. Parametric methods assume a specific distribution and seek parameters that optimize the predictive distribution by minimizing a corresponding loss. The next paragraphs introduce metrics to evaluate the predictions. Those metrics which describe both, accuracy and certainty are suitable to be used as loss function in the training process of the forecasting model.

a) Ignorance Score - Gaussian Negative Log Likelihood Loss: When forecasting distributions per time step, a forecast verification is necessary to analyze whether the true values lie within the modeled distribution. In this paper a Gaussian distribution is assumed. Its PDF is referred to as $f$, where $\mu$ denotes the mean and $\sigma$ the standard deviation, respectively. Introduced by [9], the negative logarithm of $f$, verified at the true value $y_t$ is defined as the ignorance score $\text{ign}(f, y_t) = -\log f(y_t)$, a logarithmic scoring rule that measures the success of predictions based on a modeled distribution. In the case of a normal predictive PDF with the mean $\mu$ and variance $\sigma^2$, we obtain an averaged ignorance score, here also referred to as Gaussian Negative Log-Likelihood Loss (GNLL) with

$$\text{IGN} \left[ \mathcal{N} \left( \mu, \sigma^2 \right), y \right] = \frac{1}{T} \sum_{t=1}^{T} \text{ign} (f_t, y_t)$$

$$= \frac{1}{T} \sum_{t=1}^{T} \left( \frac{1}{2} \ln \left( 2\pi\sigma^2_t \right) + \frac{(y_t - \mu_t)^2}{2\sigma^2_t} \right)$$

(1)

b) Pinball Loss - Quantile Score: A quantile $q_t$ indicates a probability $\tau$ for which the target random variable $y$ occurs within it. Given the cumulative distribution function (CDF) $F$ of $y$, the quantile is defined as $q_t = F^{-1}(\tau) = \inf \{ y : F(y) \geq \tau \}$, where $\tau \in [0, 1]$. This way, without any knowledge on the random variables’ distribution, discrete estimates of the CDF and the prediction intervals can be derived. The pinball loss quantifies the adequacy of $\hat{y}_t$, the estimate of $q_t(t)$, given the true values $y_t$, according to

$$L_\tau (y, \hat{y}) = \frac{1}{T} \left[ \tau \sum_{y_t \geq \hat{y}_t} |y_t - \hat{y}_t| + (1 - \tau) \sum_{y_t < \hat{y}_t} |y_t - \hat{y}_t| \right]$$

(2)

where $T$ is the sequence length. For $\tau < 0.5$, the estimates $\hat{y}_t$ above the true value are higher penalized and vice versa.

c) Continuous Ranked Probability Score: CRPS is the mean squared error (MSE) of the predicted CDF $F(\hat{y})$ and the Heavyside function $\mathbb{I}(\hat{y} - y)$ that is 0 if $\hat{y}_t$ is lower than the true value $y_t$ and 1 otherwise [10]. It is defined as:

$$CRPS = \frac{1}{T} \sum_{t=1}^{T} \int_{-\infty}^{+\infty} |F(\hat{y}_t) - \mathbb{I}(\hat{y}_t - y_t)|^2 d\hat{y}$$

(3)

where $T$ is the sequence length.

d) Mean Interval Score: Proposed by [11], the interval score evaluates the quality of prediction intervals. Since quantiles can be estimated with eq. (2), one can deduct the associated prediction interval (PI), which defines the range of values which can be expected to contain future observations according to a certain probability, referred to as nominal coverage rate. For a significance level $\alpha$, the coverage rate is given by $(1 - \alpha)100\%$. For determination of PIs, it is necessary to decide how they should be centered on the PDF. Commonly, the intervals are chosen to be centered on the median, such that an uncovered observed $y_t$ is equally probable to occur below or above the interval [12]. A centered PI with $\alpha$ as significance level and a nominal coverage rate of $(1 - \alpha)100\%$ is determined by using the quantile $(\hat{q}_{\tau=\alpha/2})$ as the lower bound and the $(1 - \alpha/2)$ quantile $(\hat{q}_{\tau=1-\alpha/2})$ as the upper bound, respectively. Hence, the mean interval score (MIS) averaged over the horizon $T$ is defined by

$$MIS_{\alpha} = \frac{1}{T} \sum_{t=1}^{T} \left[ (U_t - L_t) + \frac{2}{\alpha} (L_t - y_t) \mathbb{I} \left(y_t < L_t\right) \right] + \frac{2}{\alpha} (y_t - U_t) \mathbb{I} \left(y_t > U_t\right)$$

(4)

where $U_t$ and $L_t$ represent the upper and lower prediction bounds, which can be obtained either by quantiles, as stated before, or from estimated parameters like the standard deviation. For a 95%− prediction interval, $\alpha$ equals 0.05. $\mathbb{I}$ is the indicator function being 1 if $y_t$ is within the interval and 0 otherwise. Besides penalizing through the latter two terms in the equation, the first term describes the widthness of the prediction interval.

e) Prediction Interval Coverage Probability: The reliability refers to the statistical consistency between the forecasts and the observations. Its property is essential as non reliable predictions would lead to a systematic bias in subsequent
decision-making processes. To measure reliability, the Prediction Interval Coverage Probability (PICP) is given with:

$$\text{PICP} = \frac{1}{T} \sum_{t=1}^{T} c_t, \text{ where } c_t = \left\{ \begin{array}{ll} 1, & y_t \in PI^a \\ 0, & y_t \not\in PI^a \end{array} \right. \quad (5)$$

f) Sharpness: According to [12], the sharpness score is given by

$$\text{Sharpness}_a = \frac{1}{T} \sum_{t=1}^{T} \left( \hat{U}_t - \hat{L}_t \right) \quad (6)$$

A low predicted uncertainty leads to a high score and vice versa. Moreover, the score only depends on the forecast itself, not on the true observations. Thus, sharp forecasts might still be useless or even misleading in case these forecasts are off.

III. DEEP LOAD FORECASTING MODEL

A. Architecture

The deep learning models under investigation are based on recurrent neural networks (RNNs). RNNs are a natural candidate for time-series forecasting as they allow to process variable-length input sequences and facilitate weight sharing among the cells. This is achieved by keeping an internal hidden state that is fed back into the network along with the subsequent input. The underlying architecture of the considered models is illustrated in Fig. 1. It follows the encoder-decoder principle as described in [13]. The encoder is fed step-by-step with past input data $x_1(t-i)$ in order to extract the essential information and patterns and store it in the RNN cell state $h$, a fixed-length vector. The final state $h(t)$ is then passed on to the decoder. The role of the decoder is to extract the relevant information from the state and enrich it with its own inputs to pass to the prediction network, a two-layer fully-connected (FC) network with dropout. All networks are jointly trained. The RNN cell state size, the FC layer size as well as the dropout rate $p$ are subject to hyperparameter tuning. The latter, if non-zero, introduces dropout on the outputs of each layer (except the last), which randomly sets some elements of its inputs on the forward path to zero with the probability $p$. This strategy has proven to be beneficial for regularization and the prevention of neurons’ co-adaptation [14].

![Unfolded encoder](Image)

![Unfolded decoder](Image)

Fig. 1. Fundamental architecture of the encoder-decoder-based model.

In [15], the authors evaluate the adequacy of recurrent units on sequence modeling. The LSTM developed in [16], [17] is able to control its memory state by means of a forget gate. Thus, important input features can be captured over longer time frames in comparison to basic RNN cells, which tend to suffer from vanishing gradients. Similarly, the gated recurrent unit (GRU) modulates the information flow inside the unit but forgoes the introduction of a dedicated memory cell as described in [13]. The RNN cell type is also chosen through hyperparameter tuning. ReLU and leaky ReLU were chosen as activation function candidates.

To identify which common optimizer performs best, a benchmark under parameter modification of batch size and initial learning rate was conducted to compare six of the most common optimizers described in [17]–[20], including the popular Adam and AdaGrad.

The model presented in this paper is implemented with PyTorch, a deep learning framework that allows conventional, flexible python coding and debugging due to dynamic computational graphs, and also accelerates the training process by utilizing GPUs. [21]

B. Inputs

The inputs $x_1$ comprise sequential lagged data $(t-t_0, \ldots, t)$ from various different sources such as load profiles but also related explanatory variables such as meteorological data. The inputs $x_2$ comprise sequential future data $(t, \ldots, t+T)$, such as weather forecasts. Appropriate scaling of data is necessary for an efficient training process. Here, data that approximately follows a Gaussian distribution is scaled with the robust scaler using percentiles (10,90), which determines the center and scale, ignoring marginal outliers. The remaining data is scaled with the MinMax scaler on the range (-1,1) to preserve the underlying distribution. [22] The scaler parameters are determined based on the training data. Generated features (e.g. one-hot-encoding of future hours) are not subject to scaling.

C. Targets and loss function

Since the goal of this model is to predict the load and to estimate the degree of confidence in these forecasts, we can implement the introduced evaluation metrics as loss functions for training neural networks. As described before, suitable metrics are those which allow to train towards an accurate timeseries forecast connected to a corresponding confidence interval, out of which the ignorance score, the pinball loss and the CRPS are applied here. Even if the MIS is also suitable to be implemented as loss, we use it for the ex-post evaluation of the trained models only. The GNLL imposes the assumption that the target variable is approximately normally distributed. The loss describes the likelihood of the data given the two parameters. To maximize the likelihood, e.g. the GNLL is minimized during the training process, while the logarithm supports numeric stability for floating-point computation. Hence, the GNLL-loss, given in Section II-B, is used to determine the two output variables ($\mu$ and $\sigma^2$) for each time step over the forecast horizon. The QR model determines three output variables, the predicted lower and upper quantiles ($\alpha/2 = 0.025$, $1 - \alpha/2 = 0.975$) and the expected value
for each time step, using the pinball loss and the MSE, as described earlier. The CRPS-model also assumes a Gaussian distribution and computes the loss as given in (3). The output variables are the same as those of the GNLL-model.

IV. BASELINE LOAD FORECASTING MODELS

Assuming that the forecast errors are normally distributed, a 95% prediction interval for a forecast at hour \( t+h \) is \( \hat{y}_{t+h} \pm 1.96\hat{\sigma}_t \), where \( \hat{\sigma}_t \) is an estimate of the standard deviation of the forecast distribution. To produce a prediction interval, it is necessary to have an estimate of the standard deviation \( \hat{\sigma}_t \).

In order to increase interpretability and comparability of the results, we introduce two naive forecasting approaches. In our persistence forecast we assume \( \hat{y}_t \) to stay constant equal to the last observed grid state, i.e. \( \hat{y}_{t...t+40} = y_{t-1} \). Then, we estimate the standard deviation of \( \hat{\sigma}_t \) of \( e_t \) across the training data for each time step over the horizon. Instead, the periodic approach considers the previous day’s value for each corresponding hour in the forecast horizon, i.e.

\[
\mu_{t...t+40} = \begin{cases} y_{t-24} & t < 24 \\ y_{t-48} & t \geq 24 \end{cases}
\]

(7)

Both naive forecasts are exemplary illustrated in Fig. 2.

V. CASE STUDY & MODEL TESTING

The proposed model has been trained and tested with data derived from the regional distribution system in southern Sweden. Here, the load is limited with a regulated annual capacity. Especially during winter days, the area frequently experiences congestion issues. In consequence, the DSO needs to apply for expensive temporary subscription [23] (capacity and energy as shown in Fig. 3). When the increase in subscription violates constraints of the transmission system operation, the residual load has to be decreased. The forecast is intended to deliver day-ahead information on the grid load to facilitate flexibility bidding as an alternative with potentially lower expenses. The model developed in this case study generates compiled predictions each day at 9am, so that the operator can reserve extra capacity for the following day \((t+16, \cdots, t+40)\), when congestion is likely to occur. The contracted flex-capacity is typically higher than the predicted capacity shortage since losses might increase due to long distances between the area of physical flexibility activation and the congested grid point under study. In this paper, we assume that the operator aims at ensuring self-sufficient load supply with a 95% confidence. However, the presented work does not cover the optimization for subsequent flexibility trading.

A. Data Set & Pre-processing

The historical data provided covers four years of meteorological data, measured at a single station near the test site: wind direction, wind speed, rel. precipitation, and temperature. It is complemented by the measurements of the residual load aggregated over multiple primary substations, regional schedules of larger generators and cross-border exchange, each in hourly resolution. The wind power generation is rated at max. 10% of the load while the gas-fired and biomass CHP generation could cover up to 60% of the load. For the given input, time-series analysis has been performed to assess stationarity, correlation and seasonality of the residual load. The ADF-test [24], the KPSS-test [25] and the (partial) auto-correlation function reveal low stationarity, strong auto-correlation with the previous hour, and , as expected, seasonality of 24 hours. Moreover, the analysis indicates a relevant correlation with the the ambient temperature measurement (Pearson’s correlation coefficient: \( \rho \approx 0.36 \)). Also, the wind production schedule has a high impact and, naturally, reveals a negative correlation of \( \rho \approx -0.4 \). Less significant but also evident is the influence of the wind speed \( \rho \leq 0.3 \), wind direction \( \rho \approx 0.1 \), and precipitation \( \rho \approx 0.1 \). The hourly residual load over the entire period clearly shows a bell-shaped probability density curve, which justifies testing the GNLL and Gaussian CRPS. Before scaling the inputs as described in Section III-B, all missing data points (< 1% of the data) are interpolated. Finally, the data is split into a training, validation, and test set with a ratio of 0.7, 0.2, and 0.1, respectively.

B. Embedding Features

In addition to the described data set, hours of the day, months and weekdays are one-hot-encoded. Additional cyclic
features are obtained using the \( \sin \) and \( \cos \) functions applied to the hour of day to fetch continuous intraday load patterns e.g. of local energy generators to encode a a proximity of between \( h=23 \) of the previous day and \( h=0 \) of the following day. As described in Section III-A, the features are split into two input vectors \( x_1 \) (for lagged information) and \( x_2 \) (for future information), with \( t_0 = T = 40. \) \( x_1 \) is filled with the residual load, local generation and cross-border schedule data, while \( x_2 \) contains known, explanatory variables, i.e. hour, weekday, month, temperature, rel. precipitation, wind direction, and wind speed.

C. Configuration

Potential overfitting is further prevented through early stopping, where the training is interrupted in case of multiple consecutive epochs with increased validation losses. Concerning the hyperparameter tuning, a comparison of grid search, random search and minor manual tuning was conducted in [26]. Further improvement could be achieved by extending the search space or switching to other performant hyperparameter sampling algorithms. However, these optimization techniques are not examined in the presented work. At the time of writing, the best set of hyperparameters was obtained based on a combination of grid search and manual tuning, given in Tab. I.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>HYPERPARAMETER SET - GNLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best parameters and ANN configuration printed in bold</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Neurons per Layer</th>
<th>{32, 56, 112}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Layers</td>
<td>{1, 3, 5}</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>{0.0001, 0.00005, 0.00001}</td>
</tr>
<tr>
<td>Batch Size</td>
<td>{34, 50, 72, 80, 120}</td>
</tr>
<tr>
<td>Dropout Rate Cells</td>
<td>{0.3, 0.4, 0.5}</td>
</tr>
<tr>
<td>Dropout Rate Layers</td>
<td>{0.3, 0.4, 0.5}</td>
</tr>
<tr>
<td>Weight Initialization</td>
<td>{LeakyReLU ( \text{slope}=0.1 ), ReLU}</td>
</tr>
<tr>
<td>RNN Cell Type</td>
<td>{lstm, gru}</td>
</tr>
<tr>
<td>Optimizer</td>
<td>{ADAM, AdaGrad, RMSprop}</td>
</tr>
</tbody>
</table>

The machine used for training the models runs with 256 GiB RAM and two Intel Xeon Gold 6128 CPU @ 3.40GHz. The specification of the associated GPUs is given in Tab. II. On average, one training process requires 300 seconds (min. 30sec - max. 6,000sec). When using LSTM cells instead of GRU, we notice a slightly smaller MIS, mainly in consequence of a higher coverage. The LSTM-GNLL model is twice as fast compared to the GRU-GNLL, but results in a relatively higher loss (GNLL-score). While per epoch GRU is significantly faster (total computation time), the LSTM converges after less epochs. Beyond that, no further sensitivity analysis is carried regarding the extent a parameter modification eventually impacts the performance of the model.

D. Exemplary Results

A detailed example on three samples of the GNLL model is given in Fig. 4 in which the hours 0 to 24 of the following day are the ones of interest. The test samples derive from the remaining dataset which has not been used during training, covering the period Sep-2018 to Apr-2019. For the periods in which the confidence interval intersects with the horizontal line (depicting the max. capacity) it is recommended that downward flexibilities are obtained, (assuming that flex-activation is the preferred means of congestion management). As shown, the model succeeds in indicating critical grid states. Thus, the underlying Gaussian distribution with 95% confidence proves to be appropriate for weekdays, weekends, and holidays. For the sampled weekday in September, it is shown that the recurrent element can result in a systematic bias in the subsequent predictions observed for the following weekday, revealing a shortcoming of the presented model. Tab. III compares the three ANN versions and naïve predictions, measuring accuracy with RMSE [MW], certainty with sharpness [MW], and reliability with the PICP [%]. The MIS combines accuracy and certainty, indicating that the CRPS model performs best. Especially compared to the QR model, the CRPS model achieves a higher coverage, even though the certainty band is sharper. All three ANN models clearly outperform the baselines and perform equally well, while the CRPS delivers more accurate estimations but results in a smaller coverage, although targeted for a 95% prediction interval. Fig. 5 shows that the models deliver more accurate, certain and reliable predictions for shorter horizons.

VI. Conclusion

This paper presents three versions of an RNN-based probablistic forecasting model with parametric and non-parametric implementations based on Gaussian parameters and QR, respectively. Derived from a DSO’s case study, the results show that both models are suitable for a probabilistic, 40h-ahead forecast of residual loads for the purpose of future congestion indication. Preliminary studies beyond the scope of this paper indicate that the presented model is scalable and replicable. Since the forecast model is deployed in a live system, new inputs could be utilized for re-training the network. In the future, a proper re-training cycle could be analyzed and

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>GPU PERFORMANCE DATA [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tesla T4</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Single-Precision Performance (FP32)</td>
<td>8.1 TFLOPS</td>
</tr>
<tr>
<td>Integer Operations (INT8)</td>
<td>130 TOPS</td>
</tr>
<tr>
<td>GPU Memory</td>
<td>16GB</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>320GB/s</td>
</tr>
<tr>
<td>Power</td>
<td>70W</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>TOTAL AVERAGE EVALUATION SCORES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GNLL</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>63.25</td>
</tr>
<tr>
<td></td>
<td>63.78</td>
</tr>
<tr>
<td></td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>162</td>
</tr>
</tbody>
</table>

978-1-7281-2822-1/20/$31.00 ©2020 IEEE

PMAPS 2020
established. Building upon the presented results, future work could also analyze the impact of stochastic dependencies of the inputs and the sensitivity of the prediction error dependent on uncertain inputs, thus quantifying the model uncertainty and the model misspecification.

ACKNOWLEDGMENT

The authors gratefully acknowledge E.ON Energidistribu-

REFERENCES


