Paweł Szetela, Krzysztof Siwek

Warsaw University of Technology

doi: 15199/48.2024.05.51

Evaluating the forecasting capabilities of probabilistic and point-based LSTM models in sequence prediction

Abstract. *This paper compares the performance of probabilistic and non-probabilistic LSTM models in the task of univariate, real valued sequence forecasting. The performance of models is evaluated in terms of mean absolute error and root mean squared error for different forecasting horizons. The results show that probabilistic models can outperform non-probabilistic models in the task of forecasting.*

Streszczenie. *W artykule porównano wydajność probabilistycznych i nieprobabilistycznych modeli LSTM w zadaniu prognozowania szeregów czasowych. Wydajność modeli jest oceniana pod względem średniego błędu bezwzględnego i błędu średniokwadratowego dla różnych horyzontów prognozy. Wyniki pokazują, że modele probabilistyczne mogą przewyższać modele nieprobabilistyczne w zadaniu prognozowania.* ***(Ocena jakości prognostycznych modeli probabilistycznych i punktowych LSTM)***

Keywords: LSTM, RNN, probabilistic neural networks, time-series forecasting
Słowa kluczowe: LSTM, RNN, sieci neuronowe probabilistyczne, prognozowanie szeregów czasowych

Introduction

Recurrent Neural Networks (RNNs) are models designed to process temporal sequences. They implement dynamic memory in the form of recurrent connections between nodes [1]. RNNs have been used successfully for modeling short term dependencies. However, traditional RNNs poorly capture long term dependencies and are known to suffer from vanishing gradient problem, when trained with gradient based algorithms [2]. In order to eliminate vanishing gradient issue, modifications to the basic RNN architecture have been proposed, notably the Long Short-Term Memory (LSTM) [3] and Gated Recurrent Unit (GRU) [4]. Their capacity to retain information over long sequences makes them particularly suitable for forecasting tasks. Both networks have been evaluated against models most commonly used in time series forecasting (e.g. ARIMA and SVR). LSTM and GRU have achieved superior performance across multiple domains, on problems involving forecasting sequences such as energy demand, traffic flow, cryptocurrency price, stock market index, and weather, as noted in [5, 6, 7, 8, 9, 10].

Traditionally, real valued forecasting models based on RNNs have been configured for point forecasting. During training of such models, loss functions that measure distance between samples and predictions are minimized, such as mean squared error. This approach assumes constant variability in the data over time, known as homoscedasticity. Alternatively, RNNs can be configured to provide parameters of specific probability distributions, facilitating heteroscedastic, probabilistic forecasting. In addition to providing measure of uncertainty, estimated probability distributions can be used to derive point forecasts.

Probabilistic and point prediction models have been compared, in case of feedforward neural networks. Probabilistic models were demonstrated to better fit heteroscedastic data, in the task of surgery prediction [11]. RNN-based probabilistic models have been suggested for various applications, prominently in predicting power demand, traffic flow, and items sales [12, 13]. These models have demonstrated their ability to capture intricate patterns, fluctuations, and seasonal effects inherent in such datasets. However, a comprehensive comparison of these probabilistic models with their point prediction counterparts hasn't been performed, which leaves an open question regarding relative performance and advantages. Probabilistic models may mitigate error accumulation and reduced accuracy, especially in the tasks involving multistep ahead forecasting.

This research conducts a comparative analysis between point forecasting LSTM based models and their probabilistic counterparts, for real valued, univariate time series. Experiments were performed on two, real-world, publically available datasets. Performance of multistep ahead models is evaluated over selected prediction horizons. Two distinct strategies for multistep forecasting are compared: multi-input multi-output (MIMO) [14] and autoregressive LSTM models [15]. Additionally, one-step ahead model is evaluated. The core focus remains on the accuracy of point predictions. Probabilistic models predicting two probability distributions were compared: Gaussian and Laplace, commonly used for modeling real valued variables.

Initial evaluation indicates that, in terms of mean absolute error (MAE) and root mean squared error (RMSE), probabilistic models showcase a better data fit than traditional point forecasting models. This advantage in accuracy doesn't come with a trade-off in computational efficiency; both model types demand similar computational resources during their training phase.

LSTM networks

The Long Short-Term Memory (LSTM) network is one of the most widely used variant of recurrent neural networks in time series forecasting tasks. Its ability to learn and remember over long sequences has been demonstrated in numerous studies, making it a fitting choice for this research.

LSTMs process sequential data using a series of gates and states. Unlike traditional RNNs, LSTM network utilizes gating mechanism that effectively address the vanishing gradient problem, when dealing with long sequences. Additionally, LSTM network introduces 2 states that allow for capturing long and short term dependencies. For an input sequence x_t, the LSTM cell uses the input gate (i_t), forget gate (f_t), and the output gate (o_t) to update its cell state (c_t) and the hidden state (h_t) at every time step. The sigmoid function, represented by σ, outputs a value between 0 and 1, modulating content being added or removed from existing cell state. Vector of candidate values for (c_t) and (h_t) is computed by $tanh$ function, that ensures output values between -1 and 1.

The input gate determines the new information stored in the cell state:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

The forget gate decides the amount of the previous cell state to retain:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The cell state updates using the output of the forget gate and the input gate:

$$c_t = f_t \times c_{t-1} + i_t \times \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

The output gate controls the information from the cell state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(c_t)$$

Hidden state h_t is passed through a fully connected (dense) layer, which reshapes the LSTM's output to match the desired forecasting task. This adaptability enables LSTM's configuration for various forecasting strategies, both point and probabilistic.

Strategies for time series forecasting

Given a sequence of recent observations from a time series (input sequence), denoted as $y_{1:T} = \{y_1, \ldots, y_T\}$, LSTM model aims to predict future values (output sequence), denoted as $\hat{y}_{T+1:T+H} = \{\hat{y}_{T+1}, \ldots, \hat{y}_{T+H}\}$. Were T refers to the length of the input sequence and H indicates the forecasting horizon. When $H = 1$, the problem is known as one-step ahead forecasting. When $H > 1$, the model attempts to predict multiple future values, and the problem is referred to as multistep ahead forecasting. For the sake of convenience and the scope of this article, the notation assumes that the forecasts are directly subsequent to the last input time step. This means models that make lagged forecasts are not considered.

Time series models can be further categorized based on the number time-dependent variables (dimensionality of the target variable y). Univariate models forecast one variable and multivariate models forecast multiple time-dependent variables [16].

In addition to $y_{1:T}$, forecasting model is conditioned on sequence of covariates $x_{1:T}$ or $x_{1:T+H-1}$, depending on forecasting strategy employed by the model. The distinction is motivated by the assumption that variables y are known only in conditioning range $\{1, \ldots, T\}$ and covariates are inputs known across all time steps (e.g. day of the week, month of the year) [17].

Following sections describe two commonly used strategies for multistep ahead forecasting, that were employed in this research.

MIMO strategy

The first approach involves using a multi-output model that directly generates the entire sequence of future predictions in one step. Sequence of variables $y_{1:T}$ and covariates $x_{1:T}$ is processed by LSTM network. Sequence $\hat{y}_{T+1:T+H}$ is predicted directly at time step T. For this purpose, dense layer is configured to output vector of dimensionality H, corresponding to each time step within the forecasting horizon.

MIMO strategy was proposed to preserve stochastic dependency between predicted values. Incorporating the interdependencies between each time step leads to better accuracy, in comparison to models forecasting each time step in isolation. However, MIMO strategy requires fixed prediction horizon, determined by the shape of dense layer in LSTM model. Lack of flexibility in the structure prevents from reusing model in multiple forecast horizons [18].

Autoregressive strategy

Alternatively, an autoregressive strategy can be employed, which generates future predictions in an iterative manner. For these models, two temporal ranges can be distinguished. In conditioning range, at each time step $t \in \{1, \ldots, T\}$ the inputs to the LSTM model are target variables y_t and covariates x_t, which results in initial, one-step ahead prediction. In prediction range, at each time step $t \in \{T+1, \ldots, T+H-1\}$ prediction from previous time step $\hat{y}_{t-1}$ is fed back to the model, along with covariate x_t.

Providing covariates in prediction range improves computational time and accuracy, in comparison to autoregressive LSTM generating all inputs. This strategy minimizes error accumulation in long sequences [19]. Models predicting a sequence for one-dimensional variables are considered univariate, despite relying on multidimensional input vector.

Autoregressive models are known to suffer from accumulation of error for long forecasting horizons. Therefore, the accuracy of this strategy depends on level of noise present in the time series. The main advantage of autoregressive models comes from their ability to make predictions for arbitrary forecasting horizons, using single model [18].

Moving window

Training and evaluation of LSTM models often employs a rolling window strategy. This technique involves the use of consecutive, overlapping segments (windows) of the time series to train the network. An example of this process is presented in Fig. 1. Each window consists of an input

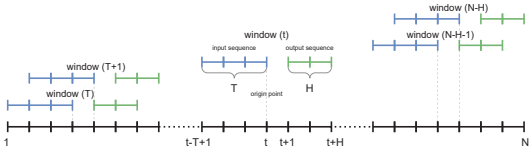

Fig. 1. Moving window example.

sequence of length T used to predict the output sequence of length H. The window then rolls forward one step for the next training iteration. For a complete time series of length N, this process results in $D = N - T - H$ pairs of input/output sequences. Utilizing overlapping windows for training recurrent model can be considered as data augmentation technique and has been shown to improve forecasting results [20]. When formulating supervised learning problem, N can denote number of time step in training, validation and test time series. Evaluating model's performance with windows (rolling analysis) gives insight into model's stability over time and allows for independent metrics for each time step in forecasting horizon [21].

Probabilistic forecasting

In many neural network-based regression problems, there exists an underlying assumption that the target's variability is global and independent of the input, leading to what is known as a homoscedastic model. Such an assumption implies that the uncertainty or noise around the predictions is constant across all levels of the independent variables. In contrast, probabilistic models that allow for input-dependent variance are categorized as heteroscedastic. These heteroscedastic models recognize that the uncertainty in predictions may vary with the input values [22]. Time series data often exhibits heteroscedasticity, where the volatility changes over time. Changing variances in time series data is well documented across multiple domains [23, 24]. Probabilistic models can better fit the data by allowing for input-dependent

variance and addressing stochastic nature of time series [11].

This distinction between homoscedastic and heteroscedastic assumptions can be crucial in the context of time series forecasting, where the nature of uncertainty can be complex and dependent on the temporal dynamics of the system. Probabilistic models allow for more nuanced understanding of the data structure, by providing uncertainty along point predictions. Assuming constant variance across time might lack robustness to capture varying uncertainty across different time steps. A homoscedastic assumption in RNNs might limit its ability to capture these dynamics, potentially affecting prediction accuracy.

Training LSTM models

In training the LSTM models, the moving window strategy was employed, where each window is rolled forward by one time step. Experiments were also conducted with non-overlapping windows, but this approach resulted in lower forecast accuracy.

The global loss function is minimized in case of all evaluated models. Loss function is defined for all training sequences as:

$$L_{global} = \sum_{i=1}^{D} L_i$$

where L_i represents the loss for the i^{th} window.

Negative likelihood loss

Likelihood function defines probability of observed data sequence, given parameters of LSTM model θ. In univariate case, it can be expressed as a product of individual sample probabilities. It is convenient to define objective function as a Negative Logarithm of Likelihood (NLL) function. Logarithm is a monotonically increasing function of its parameters and thus does not change the location of the maximum. It allows computing sum of probabilities instead of product, which is more stable numerically. Negative sign is added to convert maximization problem to minimization problem which is by convention used in optimization algorithms [25]. Likelihood functions described below provide support for real valued, single variable. LSTM models are required to estimate two parameters. For given sequence (window), NLL for Gaussian and Laplace distribution is defined as:

$$L_{i,gaussian} = \frac{1}{H} \sum_{t=T+1}^{T+H} \left(\frac{(\hat{y}_{i,t} - y_{i,t})^2}{2\hat{\sigma}_{i,t}^2} + \log(\hat{\sigma}_{i,t}) \right) \tag{1}$$

$$L_{i,laplace} = \frac{1}{H} \sum_{t=T+1}^{T+H} \left(\frac{|\hat{y}_{i,t} - y_{i,t}|}{\hat{b}_{i,t}} + \log(\hat{b}_{i,t}) \right) \tag{2}$$

In the provided equations, $\hat{y}$ is assumed to be location parameter μ of distributions. Standard deviation estimated at time t for the i^{th} sequence in (1) is denoted as $\hat{\sigma}_{i,t} > 0$. It represents the variability or dispersion of the dataset. Parameter $\hat{b}_{i,t} > 0$ in (2) indicates the estimated scale parameter, defining the spread of the distribution. The relationship between the scale parameter and the standard deviation is given by $b = 2\sigma$.

Point loss

The loss function commonly used for point forecasts in LSTM models is the mean squared error (MSE), which measures the average squared differences between the predicted values and the actual values for each time step. MSE aims at approximating the conditional mean of the target variable. MSE sequence loss can be written as:

$$L_{i,point} = \frac{1}{H} \sum_{t=T+1}^{T+H} (\hat{y}_{i,t} - y_{i,t})^2 \tag{3}$$

MSE loss can be derived from (1), under the assumption of constant variance.

Configuring LSTM models

Point LSTMs output predicted mean ($\hat{y}$) for each forecasting step. In probabilistic LSTMs, the dense output layer must produce parameters of the selected distribution. In case of Gaussian or Laplace distribution the model is tasked with estimating two parameters with similar constraints: a location parameter (predicted mean) and a positive scale parameters. Consequently, the output dimension of the dense layer in probabilistic models is twice that of point forecasting models. For each time step in the forecasting horizon, the model produces two outputs: one for the mean ($\hat{y}$) and another for the standard deviation ($\hat{\sigma}$) or the scale parameter ($\hat{b}$).

To ensure the positivity of scale parameters ($\hat{\sigma}$ and $\hat{b}$), the softplus activation function is employed in the output layer. Outputs corresponding to the mean value employ a linear activation, for probabilistic and point models.

In autoregressive LSTMs, point prediction is made for each time step in the forecasting horizon, where the prediction for one time step is used as input for predicting the subsequent time step. In the experiments, the mean of the estimated distribution was directly fed into the model as input for the next prediction.

Ancestral sampling strategy was also considered for obtaining point predictions, in autoregressive strategy. This approach involves drawing random samples from predictive distribution, at each time step in forecasting horizon, generating one sample trace. After obtaining sufficient number of sample traces, point forecasts can be estimated as median values [12]. However, this method resulted in point forecasts with marginally lower accuracy in comparison to the direct feeding of the distribution mean.

Experiments

Two publically available datasets were used for model evaluation: *electricity* and *exchange* (Table 1). Electricity dataset contains time series, logged for 370 clients (instances) [26]. Subset of 10 clients was selected randomly. Exchange dataset contains collection of the daily exchange rates of 8 foreign countries, as introduced in [27]. Since each instance can display unique patterns, separate models are trained for each instance of *electricity* and *exchange*. Datasets contain no missing values.

Table 1. Summary of the Electricity and Exchange datasets

dataset	*electricity*	*exchange*
unit	kW	exchange rate
L	35065	7588
granularity	hour	day
start time	2011-01-01	1990-01-01
instances	10	8

Original time series are split into training, validation and testing sets contain 80%, 10%, 10% of original time series, respectively. Time series are not shuffled before the

split. Variables are normalized to have mean 0 and variance 1. Mean and variance are calculated exclusively on training dataset and are used to perform reverse normalization of forecasts.

Table 2. Hyperparameters of the LSTM model

batch size	32
optimizer	Adam
learning rate	0.001
units	32
T	168
H	1, 3, 6, 12, 24, 48

Table 2 contains hyperparameters of training. Number of units refer to dimensionality of state vectors (c_t) and (h_t). Prediction horizon $h = 1$ (one step-ahead forecast) is considered as special case of autoregressive model and serves as baseline for comparing sequence error. During training, validation set loss is evaluated on each epoch. Increase in loss for two successive epochs stops the training, and weights of best performing models are preserved. Monitoring validation loss can prevent overfitting of LSTM model on training set.

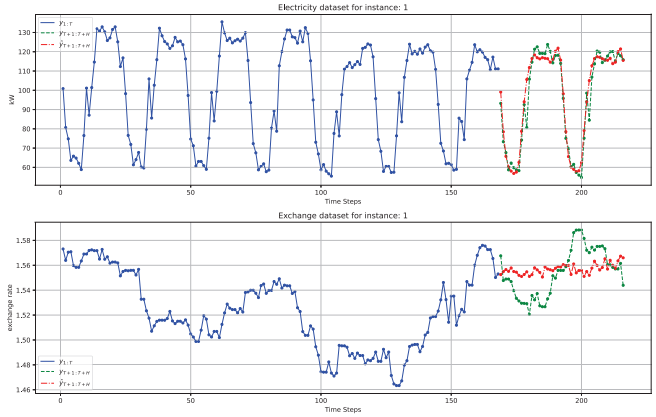

Fig. 2. Example of single window, evaluating model performance on test set (Laplace MIMO model, T = 168, H = 48).

Metrics

For evaluation of the LSTM models, a set of robust metrics has been selected to quantify their forecasting performance. Models are evaluated on test set, using moving window scheme, described in . Figure 2 shows example of single evaluation window, obtained from test sets of *electricity* and *exchange* data. Window contains samples, corresponding to forecasting horizon of 48 hours and 168 hours of historical samples. Error is evaluated in original scale, therefore, reverse normalization is performed on forecasts obtained from models, before calculating error metrics.

Introduced metrics are defined for samples at fixed forecasting horizon and are averaged over test windows, to obtain sequence error of specific forecasting step.

1. *Mean Absolute Error (MAE):* The MAE metric provides an average of the absolute forecasting errors across all observations and is calculated using the following formula:

$$MAE = \frac{1}{D}\sum_{i=1}^{D}|\hat{y}_i - y_i| \tag{4}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and N is the total number of observations.

2. *Root Mean Squared Error (RMSE):* RMSE is another popular metric used to assess the accuracy of forecasting models, especially when large errors are particularly undesirable. The RMSE can be computed as:

$$RMSE = \sqrt{\frac{1}{D}\sum_{i=1}^{D}(\hat{y}_i - y_i)^2} \tag{5}$$

3. *Percentage Reduction:* To measure the improvement or degradation in forecasting performance of probabilistic models (Gaussian and Laplace) over the point models, the percentage reduction is utilized. This metric is defined as:

$$\text{Reduction (\%)} = \left(\frac{\text{Error}_{\text{point}} - \text{Error}_{\text{distribution}}}{\text{Error}_{\text{point}}}\right) \times 100 \tag{6}$$

where $\text{Error}_{\text{point}}$ represents the error (MAE or RMSE) for the point model, and $\text{Error}_{\text{distribution}}$ is the error for either the Gaussian or Laplace model. Positive values indicate that the distribution-based model outperformed the point model, while negative values suggest the opposite.

Selected metrics encapsulate different aspects of prediction errors. MAE offers simplicity and direct interpretability; each error impacts MAE directly in proportion to its absolute magnitude. RMSE, by squaring the differences, brings focus to larger errors, causing them to have bigger influence on the final value. Consequently, a few substantial deviations can elevate RMSE significantly more than MAE. This differential treatment of errors highlights that RMSE is more sensitive to outliers, whereas MAE provides a linear error perspective [28].

Results

Performance of LSTM models was compared in similar training setting. Combination of LSTMs employing different forecasting strategies (*one-step*, *autoregressive*, *one-step*) and minimizing different objectives (*point*, *gaussian*, *laplace*) was considered. Models were trained independently, for each instance of datasets. Table 3 reports performance figures averaged across 10 randomly selected clients. Table 4 reports performance figures averaged across exchange series for 8 available countries. The best results for MAE and RMSE metrics are marked in bold (lower is better).

Laplace models reliably outperform other methods, except in autoregressive strategy for electricity dataset. One-step LSTMs minimizing Laplace objective achieved the lowest MAE and RMSE for both datasets.

Gaussian distribution proved to describe data worse than Laplace distribution. Models performed notably worse in MIMO and one-step strategy in electricity dataset, in comparison to Laplace and point models.

Long forecasting horizons in autoregressive strategy tend to lower the reduction of error of probabilistic models. Additionally, performance of autoregressive probabilistic models, relative to point models, shows high variability across forecasting horizons. This is most notable in table 4 for Laplace model, where reduction of RMSE and MAE exceeds 20% for forecasting horizon H = 3, while for H = 48, increase of error exceeding 9 % is observed.

MIMO strategy results in lower error than autoregressive.

Table 3. One-step vs. autoregressive vs. mimo models for electricity dataset, averaged across instances

Strategy		one-step	autoregressive					mimo				
Horizon		1	3	6	12	24	48	3	6	12	24	48
3*MAE	point	23.680	28.366	32.837	36.174	**33.818**	38.910	27.715	31.883	34.356	34.228	36.576
	gauss	24.599	27.836	**30.250**	34.956	35.020	**37.854**	27.859	33.181	35.685	34.703	37.176
	laplace	**22.234**	**27.730**	31.136	**32.110**	34.503	39.149	**27.245**	**29.851**	**31.731**	**31.645**	**35.422**
2*Reduction (%)	gauss	-3.88	1.87	7.88	3.36	-3.56	2.72	-0.52	-4.07	-3.87	-1.39	-1.64
	laplace	6.11	2.24	5.18	11.23	-2.03	-0.61	1.70	6.37	7.64	7.55	3.16
3*RMSE	point	33.311	39.459	45.267	52.337	**48.594**	54.759	**38.460**	44.630	47.928	48.834	51.863
	gauss	35.055	39.287	**42.862**	50.212	50.299	**54.177**	39.426	45.809	50.365	49.108	52.743
	laplace	**32.955**	**39.103**	44.343	**47.438**	50.934	55.728	38.974	**42.772**	**45.838**	**46.232**	**51.268**
2*Reduction (%)	gauss	-5.24	0.44	5.31	4.06	-3.51	1.06	-2.51	-2.64	-5.09	-0.56	-1.70
	laplace	1.07	0.90	2.04	9.36	-4.81	-1.77	-1.34	4.16	4.36	5.33	1.15

Table 4. One-step vs. autoreggresive vs. mimo models for exchange dataset, averaged across instances

Strategy		one-step	autoregressive					mimo				
Horizon		1	3	6	12	24	48	3	6	12	24	48
3**MAE	point	3.822	6.517	7.513	11.025	14.687	20.835	5.890	7.503	9.294	12.433	16.352
	gauss	3.551	5.620	8.073	11.771	14.482	**20.281**	5.540	7.124	9.013	**11.199**	15.809
	laplace	**3.488**	**5.144**	**7.302**	**10.766**	**14.378**	22.928	**5.181**	**6.686**	**8.806**	11.596	**14.059**
2*Reduction (%)	gauss	7.07	13.77	-7.44	-6.77	1.40	2.66	5.94	5.05	3.02	9.93	3.32
	laplace	8.73	21.08	2.81	2.35	2.10	-10.04	12.03	10.89	5.24	6.73	14.02
3**RMSE	point	5.157	8.775	9.930	14.486	19.249	26.419	7.790	9.887	12.182	16.124	20.812
	gauss	4.879	7.599	10.693	15.360	19.645	**26.357**	7.601	9.592	11.814	**14.453**	19.872
	laplace	**4.833**	**6.850**	**9.688**	**14.158**	**18.533**	28.853	**6.998**	**8.832**	**11.396**	14.714	**17.876**
2*Reduction (%)	gauss	5.40	13.39	-7.68	-6.03	-2.05	0.23	2.42	2.98	3.02	10.37	4.51
	laplace	6.28	21.94	2.44	2.26	3.72	-9.21	10.17	10.67	6.45	8.75	14.11

*$1e-3$ units for readability.

Levels of error reduction are consistent across forecasting horizons. Laplace MIMO models resulted in the lowest MAE and RMSE in 4 out of 5, or all forecasting horizons, in both datasets.

Reduction of MAE and RMSE was maintained at similar levels across forecasting horizons, for both datasets. This indicates that probabilistic models do not affect outliers in disproportionate way, in comparison to point models.

It is worth noting that increasing forecasting horizons results in increase of MAE and RMSE for all models. No strategy or objective function was able to mitigate accumulation of error in long sequences. Additionally, MIMO strategy resulted in lower error than autoregressive, independently of objective function used.

Conclusion

Performed experiments showed that, aside from modeling uncertainty of point predictions, probabilistic models can reliably reduce forecasting error. The choice of probability distribution is crucial for the performance of probabilistic models and depends on dataset being modeled. Set of NLL functions, applicable for given problem, can be considered as hyperparameter when training probabilistic models.

Comparison can be further extended to univariate cases other than real valued variables. For example, impact of loss function on forecasting categorical and ordinal variables can be evaluated.

Experiments were performed on univariate datasets. However, the proposed approach can be extended to multivariate time series. In such case, multivariate distributions can be used to model dependencies between variables. Formulating optimization problem can be more complex, when accommodating marginal distributions and their dependencies (e.g. copula functions).

***Authors**: Paweł Szetela, Krzysztof Siwek, Ph.D. D.Sc. prof. PW, Institute of Theory of Electrical Engineering, Measurement and Information Systems, Faculty of Electrical Engineering, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland, email: Krzysztof.Siwek@pw.edu.pl*

REFERENCES

[1] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[5] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2018.

[6] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2016, pp. 324–328.

[7] S. McNally, J. Roche, and S. Caton, "Predicting the price of bitcoin using machine learning," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2018, pp. 339–343.

[8] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1394–1401.

[9] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The perfor-

mance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 3285–3292.
[10] Y. Yu, J. Cao, and J. Zhu, "An lstm short-term solar irradiance forecasting under complicated weather conditions," *IEEE Access*, vol. 7, pp. 145 651–145 666, 2019.
[11] N. Ng, R. A. Gabriel, J. McAuley, C. Elkan, and Z. C. Lipton, "Predicting surgery duration with neural heteroscedastic regression," 2017.
[12] V. Flunkert, D. Salinas, and J. Gasthaus, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *CoRR*, vol. abs/1704.04110, 2017. [Online]. Available: http://arxiv.org/abs/1704.04110
[13] Y. Zhang, J. Wang, and X. Wang, "Review on probabilistic forecasting of wind power generation," *Renewable and Sustainable Energy Reviews*, vol. 32, pp. 255–270, 2014.
[14] G. Bontempi, "Long term time series prediction with multi-input multi-output local learning," *Proceedings of the 2nd European Symposium on Time Series Prediction (TSP), ESTSP08*, 01 2008.
[15] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850
[16] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020300996
[17] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20200209, 2021. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2020.0209
[18] S. B. Taieb, G. Bontempi, A. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition," 2011.
[19] M. W. Seeger, D. Salinas, and V. Flunkert, "Bayesian intermittent demand forecasting for large inventories," *Advances in Neural Information Processing Systems*, vol. 29, 2016.
[20] S. Smyl and K. Kuber, "Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks," 07 2016.
[21] E. Zivot and J. Wang, *Modeling Financial Time Series with S-PLUS®*, ser. International Federation for Information Processing. Springer New York, 2007. [Online]. Available: https://books.google.pl/books?id=sxODP2l1mX8C
[22] P. Goldberg, C. Williams, and C. Bishop, "Regression with input-dependent noise: A gaussian process treatment," *Advances in Neural Information Processing Systems*, vol. 10, 02 1998.
[23] V. Akgiray, "Conditional heteroscedasticity in time series of stock returns: Evidence and forecasts," *The Journal of Business*, vol. 62, no. 1, pp. 55–80, 1989. [Online]. Available: http://www.jstor.org/stable/2353123
[24] B. Whitcher, S. Byers, P. Guttorp, and D. Percival, "Testing for homogeneity of variance in time series: Long memory, wavelets and the nile river," *Water Resour. Res.*, vol. 38, 06 1999.
[25] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.
[26] A. Trindade, "ElectricityLoadDiagrams20112014," UCI Machine Learning Repository, 2015, DOI: https://doi.org/10.24432/C58C86.
[27] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," 2018.
[28] R. Pontius, O. Thontteh, and H. Chen, "Components of information for multiple resolution comparison between maps that share a real variable," *Environmental and Ecological Statistics*, vol. 15, pp. 111–142, 06 2008.