Hyper-Parameter Optimization of Multi-Attention Recurrent Neural Network for Battery State-of-Charge Forecasting*

Aleksei Mashlakov[0000−0001−7370−3805], Ville Tikka[0000−0002−6395−3373], Lasse Lensu[0000−0002−7691−121X], Aleksei Romanenko[0000−0003−0814−954X], and Samuli Honkapuro[0000−0001−8761−474X]

LUT University, Yliopistonkatu 34, 53850 Lappeenranta, Finland
{aleksei.mashlakov, ville.tikka, lasse.lensu, aleksei.romanenko, samuli.honkapuro}@lut.fi

Abstract. In the past years, a rapid deployment of battery energy storage systems for diverse smart grid services has been seen in electric power systems. However, a cost-effective and multi-objective application of these services necessitates a utilization of forecasting methods for a development of efficient capacity allocation and risk management strategies over the uncertainty of battery state-of-charge. The aim of this paper is to assess the tuning efficiency of multi-attention recurrent neural network for multi-step forecasting of battery state-of-charge under provision of primary frequency control. In particular, this paper describes hyper-parameter optimization of the network with a tree-structured parzen estimator and compares such optimization performance with random and manual search on a simulated battery state-of-charge dataset. The experimental results demonstrate that the tree-structured parzen estimator enables 0.6% and 1.5% score improvement for the dataset compared with the random and manual search, respectively.

Keywords: battery state-of-charge · hyper-parameter optimization · multi-attention neural network · random search · tree parzen estimator.

1 Introduction

Battery energy storage systems (BESSs) are expected to be one of the key smart grid technologies residing in all electric grid levels and providing variety of system- or grid-oriented services [15, 11]. Primary frequency control (PFC) with a fast dynamic response is one of the lucrative and highest-value application of BESSs from a power system stability and economic perspectives [16, 3]. However, many studies conclude that in order to realize the full potential of costly batteries, it requires a multi-objective operation and hence estimation of how its state-of-charge (SOC) capacity will change in different time intervals [10]. Consequently, forecasting methods are of extraordinate importance for the optimal utilization of BESSs in the multi-objective smart grid environment.

* Supported by the DIGI-USER research platform of LUT University, Finland.

Time-series forecasting with deep learning has proven to be efficient tool for a time-dependent decision making in electric power systems. Most of the applications found in the literature can be related to forecasting of wind power generation [9], load consumption [13], market price [24], and solar photo-voltaic (PV) generation [14], respectively. There is, however, a deficit of expertise in forecasting of BESS SOC under the PFC, and the existing research in related domains is limited by rare attempts of frequency deviation forecasting [8].

This paper aims to assess the efficiency of multi-attention recurrent neural network (MARNN) to forecast hourly BESS SOC delta under the PFC on multi-step time intervals. However, the performance of such deep learning models is highly dependent on the selection of model hyper-parameters and, hence, requires comprehensive hyper-parameter tuning prior to an evaluation of the model efficiency. In order to solve this hyper-parameter optimization problem, a tree-structured parzen estimator (TPE) introduced in [6, 5] is deployed for the MARNN and described in this paper. Moreover, the results of the paper provide a comparison of the performance of the TPE optimization with a random and manual search. The validation of the optimization approaches is carried out on a simulated BESS SOC dataset generated based on the historical frequency data measured in continental Europe synchronous area.

This paper is structured as follows. Section 2 provides a general overview of the hyper-parameter optimization and the TPE. The description of an applied automatic hyper-parameter tuning methodology including testing dataset, neural network model, hyper-parameter search spaces, and scenarios are given in Section 3. The performance evaluation of the results is presented in Section 4. Finally, discussion of the results and conclusions are given in Sections 5 and 6, respectively.

2 Background

2.1 Hyper-parameter optimization

Hyper-parameter optimization in machine learning assesses the problem of finding a set of optimal hyper-parameters x^* in the domain χ that return the best performance as evaluated on a validation set x:

$$x^* = \arg\min_{x \in \chi} f(x), \tag{1}$$

where the best score is defined as minimum of objective function $f(x)$ that usually corresponds to an error rate or a loss function.

Most of the hyper-parameter optimization techniques can be categorized to manual, grid search, random search, and Bayesian model-based optimization in the increasing order of their efficiency [4]. The highest efficiency of the Bayesian approaches can be explained by taking into account the results of previous evaluations in contrast to the grid and the random search. The efficiency of Bayesian optimization is estimated with a probability model of the objective function based on observed hyper-parameter values:

$$P(f(x)|x). \tag{2}$$

This probability model is called a "surrogate" and represents a high-dimensional response surface of hyper-parameters mapped to the probability of a score on the objective function. Hence, the hyper-parameters in Bayesian optimization are chosen based on greater probability of surrogate model and then evaluated on the actual objective function.

2.2 Tree Parzen Estimator

The Tree-structured Parzen Estimator is a variant of sequential Bayesian model-based optimization. Similar to the other model-based optimizations, in TPE every hyper-parameter has a domain (search space) that is expressed via the probability distributions such as uniform, log-normal, and normal distributions, or categorical variables. However, dissimilar to other algorithms, for each hyper-parameter, the TPE creates two different distributions, where $l(x)$ is a distribution for the hyper-parameters where the value of the objective function is less than the threshold y^*, and $g(x)$ is the distribution that is greater than the threshold. Then, the TPE uses a Bayes rule to build a surrogate model with the probability of the hyper-parameters given the score on the objective function:

$$P(x|f(x)) = \begin{cases} l(x), & \text{if } y < y^* \\ g(x), & \text{if } y \geq y^* \end{cases}. \tag{3}$$

Finally, the next set of hyper-parameters is selected from the surrogate model with aim to maximize Expected Improvement (EI) criteria that is proportional to $l(x)/g(x)$ ratio and promotes a choice of hyper-parameters from $l(x)$ distribution:

$$EI_{y^*}(x) = \frac{\gamma y^* l(x) - l(x) \int_{y^*}^{-\infty} p(y)dy}{\gamma y^* l(x) + (1-\gamma) g(x)} \propto (\gamma + \frac{g(x)}{l(x)}(1-\gamma))^{-1}, \tag{4}$$

where γ is a quantile of the observed y values, so that $\gamma = P(y < y^*)$.

3 Methods

3.1 Automatic hyper-parameter tuning

The process of automatic hyper-parameter tuning applied in this paper for the TPE optimization is described in Fig. 1 as sequential Bayesian model-based optimization. This automatic process can be explained as closed loop simulation between the model of a neural network and Bayesian optimizer. The part of the neural network simulation starts with building the neural network model on a set of hyper-parameters from search space, continues with the network training and the evaluation of the network performance on a validation set. A history of such evaluations serves as an input for the optimizer that is using this history to form a surrogate model and obtain new hyper-parameters maximizing EI criteria. This iterative process is repeated until the maximum number of estimations is reached. In the case of random search, such process does not assume any evaluation of the previous results and generates the hyper-parameter set randomly.

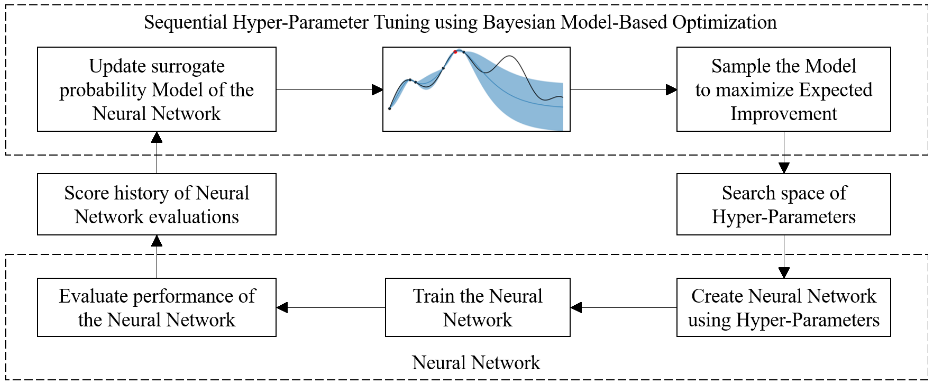
Sequential Hyper-Parameter Tuning using Bayesian Model-Based Optimization
Update surrogate probability Model of the Neural Network
Sample the Model to maximize Expected Improvement
Score history of Neural Network evaluations
Search space of Hyper-Parameters
Evaluate performance of the Neural Network
Train the Neural Network
Create Neural Network using Hyper-Parameters
Neural Network

Fig. 1. A sequential automatic hyper-parameter optimization modified from [17].

3.2 Testing data

The BESS SOC dataset for the automatic hyper-parameter tuning was modelled according to the rules for provision of the PFC that are based on the deviation of locally measured frequency $f(t)$ from the nominal system frequency f_{N}:

$$\Delta f(t) = f(t) - f_{\mathrm{N}}. \tag{5}$$

A power output at every moment $P_{\mathrm{FCR}}(t)$ required for the PFC is defined by a reference droop curve that is shaped by two main parameters, which are an allowed dead-band Δf_{db} and a full activation frequency deviation $\Delta f_{\max}$:

$$P_{\mathrm{FCR}}(t) = \begin{cases} 0, & |\Delta f(t)| \leq |\Delta f_{\mathrm{db}}| \\ P_{\mathrm{FCR}}^{\max}\left(\frac{\Delta f(t)}{|\Delta f_{\max}|}\right), & |\Delta f_{\mathrm{db}}| < |\Delta f(t)| < |\Delta f_{\max}| \\ P_{\mathrm{FCR}}^{\max}\left(\frac{\Delta f(t)}{|\Delta f(t)|}\right), & |\Delta f(t)| \geq |\Delta f_{\max}| \end{cases}. \tag{6}$$

A negative frequency deviation over the dead-band corresponds to a positive reference power output, which in the case of BESS leads to discharging, while BESS charging is provoked by the positive deviation under the same condition. A BESS is in idle state when the frequency deviation is within the dead-band. When the frequency deviations are larger that the dead-band, BESS regulation power is proportionally increased until the full activation frequency limit is reached. Finally, the deviations that are equal or exceeding $\Delta f_{\max}$ require a maximum reference power. Besides these parameters there is a full activation time for the resources, but BESS can provide continuous support with a small activation time.

Historical grid frequency measurements with 10-second resolution retrieved from the Rseau de Transport d'lectricit (RTE - transmission system operator of France) [21] were utilized as an input parameter for the dataset modelling. The measurements correspond to a continental Europe synchronous area during the time period from October 2014 to April 2019. The droop curve $\Delta f_{\max}$ was set to ± 200 mHz with no-activation deadband Δf_{db} of ± 10 mHz. A chosen BESS power to energy ratio was equal to 1 as it is one of the most common ratios for PFC according to [11]. The resulted 10-second BESS SOC dataset has been re-sampled to a hourly SOC resolution and is depicted in Fig. 2. The data is stationary with a vivid correlation at 24-hour lagged data points that is illustrated in Fig. 3.

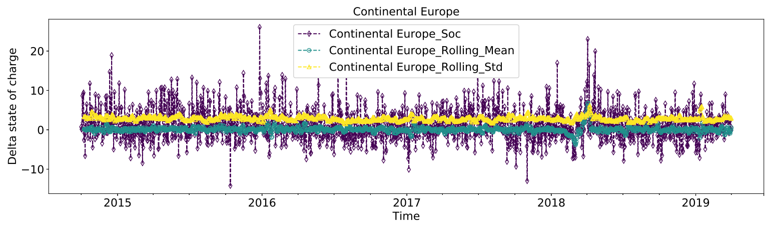

Fig. 2. Raw, standard deviation, and rolling mean values of BESS SOC dataset.

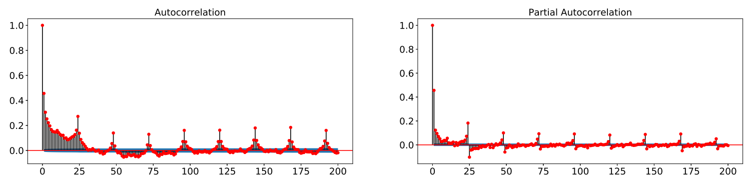

Fig. 3. Autocorrelation and partial autocorrelation plots of BESS SOC dataset with 200-hour lag values.

3.3 Model description

Recurrent Neural Networks (RNNs) that were initially developed for language modelling [23] are one of the state-of-art architecture designs applied for solving sequential forecasting problems in an energy sector [22, 18]. A structure of RNN models usually consists of encoder and decoder RNNs represented by a set of long short term memory (LSTM) units or gated recurrent unit (GRU). In this architecture, the former is trying to compress an input sequence into a context vector and the latter attempts to decode this vector in an output sequence.

Attention mechanism is one of the latest advances in neural machine translation [2] that led to significant performance improvements of deep learning models. The main difference with the encoder-decoder RNN is that the attention model develops an aggregate context vector that is filtered specifically for each output time step and memorized in the decoder layer. In this work, MARNN is deployed to assess its effectiveness on the above introduced dataset. A MARNN was implemented based on the model in [20] using Keras 2.0.2 high-level neural networks API [7] with Tensorflow 1.0.1 [1] as backend in the Python 3.6 environment, and an example functional model of this network with two attention heads is visualized in Fig. 4. This functional model can be separated into Input, Encoder, Attention, and Decoder layers.

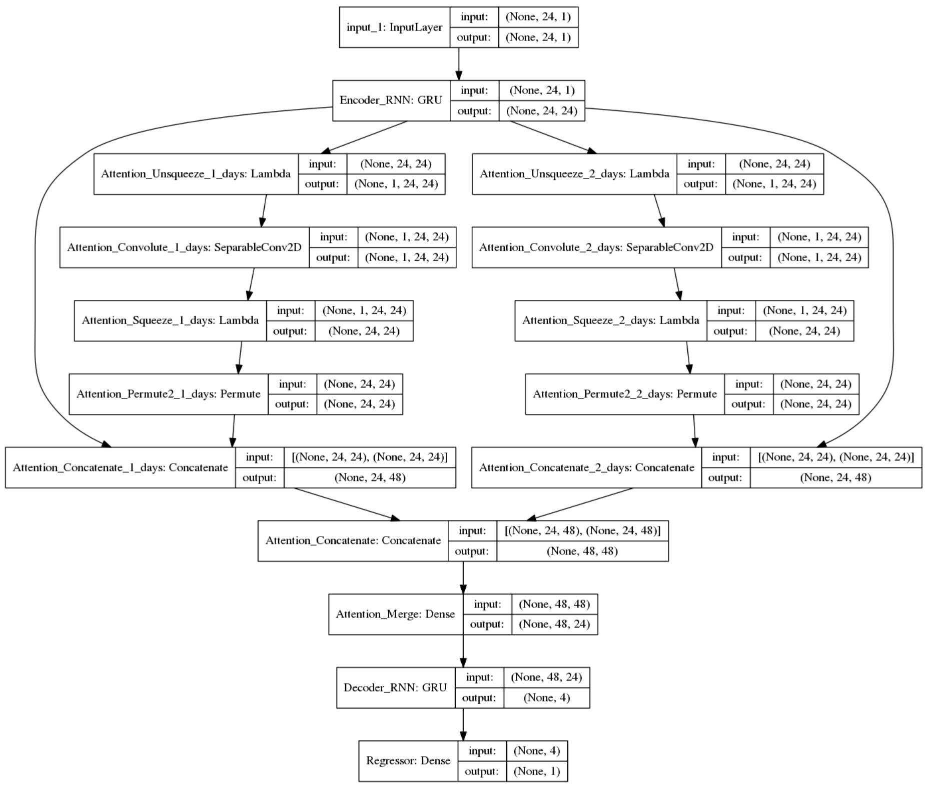
input_1: InputLayer | input: (None, 24, 1) | output: (None, 24, 1)
Encoder_RNN: GRU | input: (None, 24, 1) | output: (None, 24, 24)
Attention_Unsqueeze_1_days: Lambda | input: (None, 24, 24) | output: (None, 1, 24, 24)
Attention_Unsqueeze_2_days: Lambda | input: (None, 24, 24) | output: (None, 1, 24, 24)
Attention_Convolute_1_days: SeparableConv2D | input: (None, 1, 24, 24) | output: (None, 1, 24, 24)
Attention_Convolute_2_days: SeparableConv2D | input: (None, 1, 24, 24) | output: (None, 1, 24, 24)
Attention_Squeeze_1_days: Lambda | input: (None, 1, 24, 24) | output: (None, 24, 24)
Attention_Squeeze_2_days: Lambda | input: (None, 1, 24, 24) | output: (None, 24, 24)
Attention_Permute2_1_days: Permute | input: (None, 24, 24) | output: (None, 24, 24)
Attention_Permute2_2_days: Permute | input: (None, 24, 24) | output: (None, 24, 24)
Attention_Concatenate_1_days: Concatenate | input: [(None, 24, 24), (None, 24, 24)] | output: (None, 24, 48)
Attention_Concatenate_2_days: Concatenate | input: [(None, 24, 24), (None, 24, 24)] | output: (None, 24, 48)
Attention_Concatenate: Concatenate | input: [(None, 24, 48), (None, 24, 48)] | output: (None, 48, 48)
Attention_Merge: Dense | input: (None, 48, 48) | output: (None, 48, 24)
Decoder_RNN: GRU | input: (None, 48, 24) | output: (None, 4)
Regressor: Dense | input: (None, 4) | output: (None, 1)

Fig. 4. An example of a functional model of the MARNN with two attention heads and four decoder units.

3.4 Hyper-parameters

Attention length corresponds to a number of multi-attention heads used in the attention layer and defines possible capability of the attention mechanism. *Num-*

ber of hidden units in the decoder layer specifies the number of states kept in the decoder layer and is tightly linked with the attention length. For instance, setting the number to 4 units in case of 24-hour input sequence will provide only 4 days saved for the attention at maximum. *Activation function* defines the relevance of the attention by leveling the output of aggregated context vector in a concatenated part of the attention layer. *Dropout* addresses an over-fitting problem by randomly dropping units from the neural network during training. In the model, the dropout was applied to a densely-connected attention and decoder layers, respectively. *Learning rate* is one of the crucial parameters for training. Usually it varies from less than 1 to 10^{-6} with 10^{-2} as default value. In the model, the learning rate was adjusted for Adam optimizer.

3.5 Testing scenario

A testing scenario applied in this study for the automatic hyper-parameter tuning is presented in Table 1. The parameters of the testing scenario are defined by search spaces and distributions of the above-described hyper-parameters. The search spaces for the number of hidden units and attention heads were set in range from 2 to 14 with randint distribution. Categorical variable choice was used for the activation function to select among None, ReLu, and Sigmoid functions. Finally, an uniform distribution was used for the dropout and the learning rate with corresponding ranges from 0 to 1 and from 10^{-3} to 10^{-2}, respectively.

Table 1. Scenarios for a hyper-parameter optimization of the model.

Hyper-parameter	Search space	Distribution
Attention length	$2 - 14$	Randint
Hidden units	$2 - 14$	Randint
Activation function	None, Sigmoid, ReLu	Categorical
Dropout 1	$0.0 - 1.0$	Uniform
Dropout 2	$0.0 - 1.0$	Uniform
Learning rate	$10^{-3} - 10^{-2}$	Uniform

In this work, the automatic hyper-parameter optimization was implemented with Hyperas package [19] that is a wrapper over Hyperopt library [12]. The objective function for the performance evaluation of the MARNN model was Root Mean Square Error (RMSE). Prior to the testing, a difference was applied to the dataset to prevent persistence model properties. Moreover, a MinMax scaling with range from 0 to 1 was utilized for the dataset. A number of maximum iterations was limited to 100 for both algorithms, and a number of epochs was limited to 10 for each trial. A batch size was set equal to 128 data points. Two optimization approaches were used from the Hyperas, which are the TPE and random search.

4 Results

A visualization of the automatic hyper-parameter optimization with the TPE and the random search is presented in Fig. 6-5 with PairGrid plot and the best optimization parameters are summarized in Table 2 including the optimal manual search parameters. The color of the points in a scatter plot above a diagonal is mapped to a sequential order of the testing trials, where dark blue points correspond to the first trials and dark red to the last trials. A density plots below the diagonal demonstrate the highest concentration of the trials with dark red color and the lowest with dark blue color. The best trial is marked with a red star, and diagonal plots show the distribution of the trials along the search spaces. In both of the methods, some of the testing estimations were removed from the visualization because of the extremely high values of their RMSE scores. In Table 2, the best RMSE score among the methods is marked in bold.

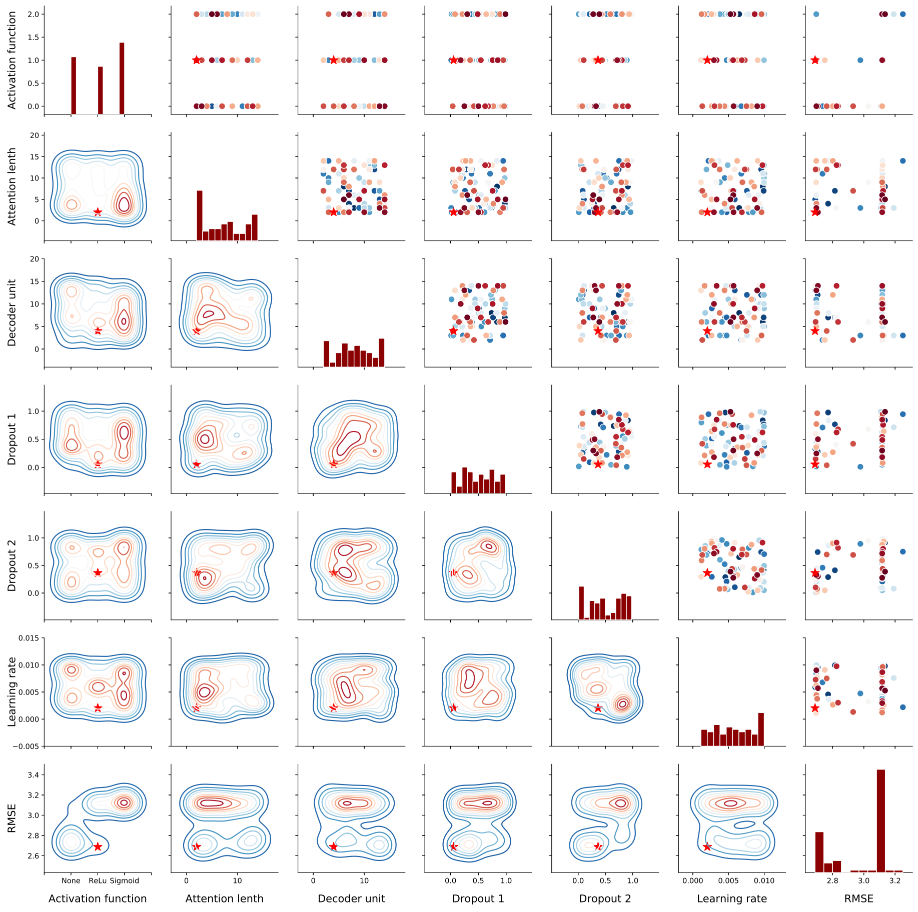

Fig. 5. Results of the random search hyper-parameter optimization.

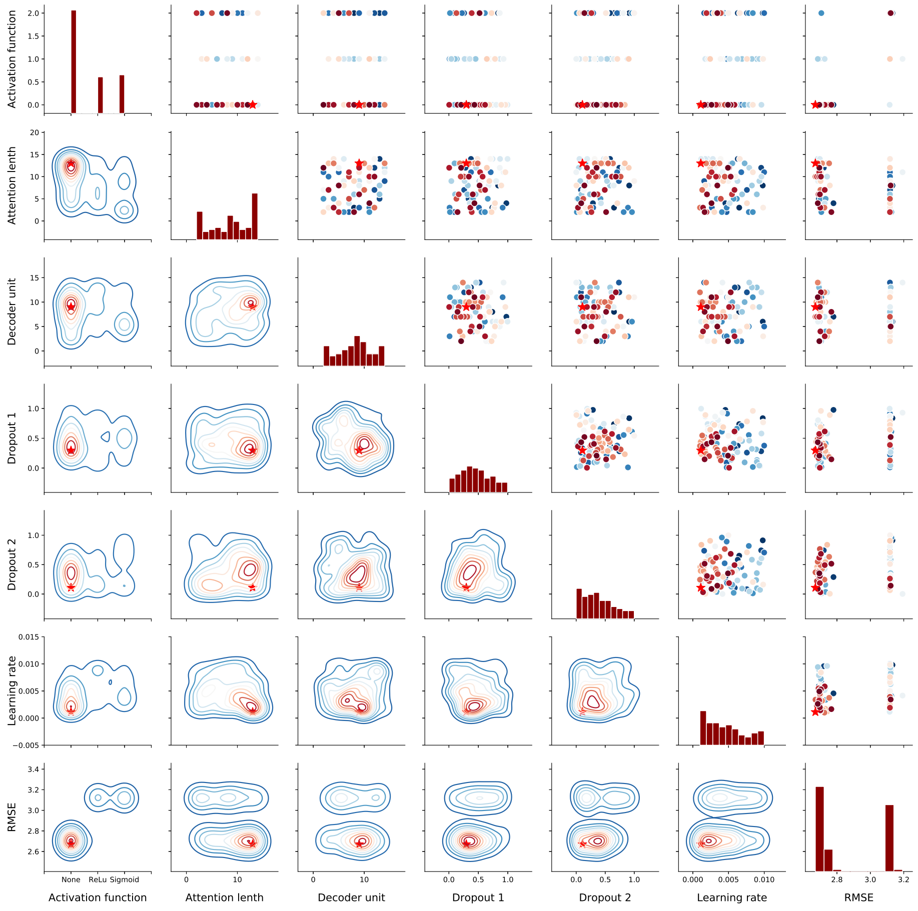

Fig. 6. Results of the TPE hyper-parameter optimization.

Table 2. Results of the model hyper-parameter optimization.

Hyper-parameter	TPE optimization	Random search	Manual search
Attention length	13	2	7
Hidden units	9	4	8
Activation function	None	ReLu	None
Dropout 1	0.295	0.054	0.2
Dropout 2	0.108	0.364	0.2
Learning rate	0.001	0.002	0.01
RMSE	**2.671**	2.688	2.710

According to the results, the best score was achieved by the TPE method with a difference from random and manual search approaches in 0.6% and 1.5%, respectively. However the scores of all methods are comparable, the hyper-parameters used for their best trials have major dissimilarities. For example, the TPE optimization achieved its best result with None activation function, while ReLU was the best choice for the random search. The TPE was searching for the attention length and the number of decoder units near the highest values of the given search space, while just 2 multi-attention heads and 4 decoder units were used in the random search. This dissimilarity can be also seen from the dropout values where a midpoint area of the interval was used for the dropout 1 and the beginning - for the dropout 2 by the TPE, but the opposite combination was utilized in the random search. However, a partial convergence can be seen in low values for the learning rate in both cases.

A performance of the tested methods is demonstrated in Fig. 7 on 24-hour forecasting intervals that were chosen randomly from the testing set. For each of the method, the MARNN was built based on the best trial hyper-parameters, trained during 50 epochs, and tested against BESS SOC testing data.

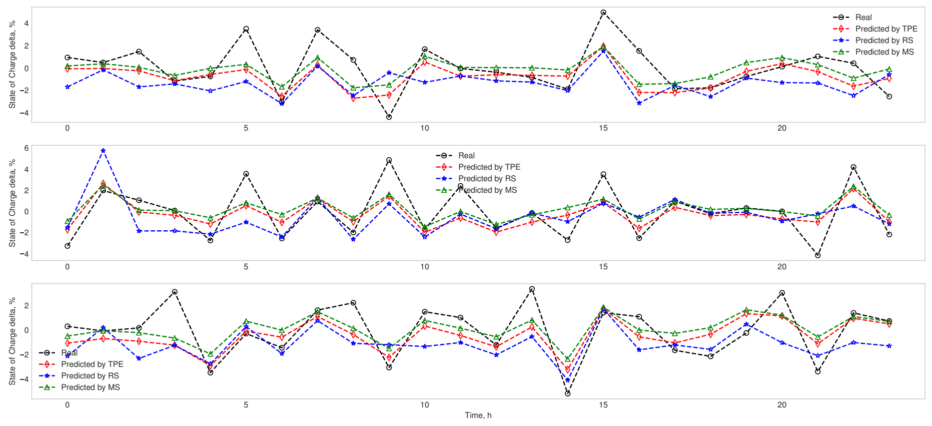

Fig. 7. A performance of the TPE, random search, and manual search optimization methods on unobserved testing data.

5 Discussion

The outcomes of the automatic hyper-parameter tuning are inline with the results shown by these optimization algorithms in other studies. The advantages of TPE are in lower number of iterations required for a score improvement compared with the random search and generally higher score performance. However, in this case, the TPE optimization has not improved the neural network accuracy notably, it gave much more understanding about areas with the highest expected improvement. It is also possible that the best results were not achieved because of the low number of testing trials that might not have been sufficient for TPE to select the best hyper-parameters. However, a higher number of trials was not possible due to hardware memory limits. In this situation, a possible approach would be an iterative simulations with several testing search spaces that would be chosen from the most promising areas of each previous optimization test. Moreover, taking into account the similarity of the models results and the different hyper-parameters values used for the different methods, it is possible to conclude that the obtained performance scores are close to the optimal value, and further performance improvement is restricted by the effects of the chosen hyper-parameters on the model score. Moreover, an additional limitation of the utilized approach is that the methods were delivering the best score of each trial based on the last epoch but it is also possible that the lowest score could have been achieved at early epochs. Moreover, a better approach would be to substitute the restriction of fixed number of epochs with early-stopping criteria and to save the best score among the epochs.

6 Conclusions

This paper describes a deployment of the automatic hyper-parameter tuning on top of the MARNN model for forecasting of BESS SOC under the PFC. The TPE optimization was applied for hyper-parameter tuning and compared with the random and manual search. The best score was achieved by the TPE but the close performances were demonstrated by the random and manual search. Taking into consideration the similarity of the results with different hyper-parameters, it is possible to assume the near-optimal values of the chosen hyper-parameters for BESS SOC forecasting.

The future work should resolve the limitations of the current work and investigate the results of TPE optimization for a larger number of trials with different stopping criteria for the trials. Moreover, an assessment of the TPE optimization efficiency for BESS SOC datasets from grid synchronous areas with different provision curve parameters is a matter of interest.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 265–283 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Belonogova, N., Tikka, V., Haapaniemi, J., Haakana, J., Honkapuro, S., Partanen, J., Heine, P., Pihkala, A., Hellman, H.P., Hvvarinen, M.: Methodology to define a BESS operating strategy for the end-customer in the changing business environment. In: 2018 15th International Conference on the European Energy Market (EEM). pp. 1–6. IEEE (2018)
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(Feb), 281–305 (2012)
5. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures (2013)

6. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems. pp. 2546–2554 (2011)
7. Chollet, F., et al.: Keras. https://keras.io (2015)
8. Chourey, D.R., Gupta, H., Kumar, A., Kumar, J., Kumar, A., Mishra, A.: Analyzing effect of system inertia on grid frequency forecasting usnig two stage neuro-fuzzy system. Journal of The Institution of Engineers (India): Series B **99**(2), 125–136 (2018)
9. Dong, D., Sheng, Z., Yang, T.: Wind power prediction based on recurrent neural network with long short-term memory units. In: 2018 International Conference on Renewable Energy and Power Engineering (REPE). pp. 34–38. IEEE (2018)
10. Fitzgerald, G., Mandel, J., Morris, J., Touati, H.: The economics of battery energy storage: How multi-use, customer-sited batteries deliver the most services and value to customers and the grid. Rocky Mountain Institute p. p6 (2015)
11. Hesse, H., Schimpe, M., Kucevic, D., Jossen, A.: Lithium-ion battery storage for the grid a review of stationary battery storage system design tailored for applications in modern power grids. Energies **10**(12), 2107 (2017)
12. Hyperopt: Distributed asynchronous hyperparameter optimization in Python. (2012–), http://hyperopt.github.io/hyperopt/, [Online; accessed 30-04-2019]
13. Kong, W., Dong, Z.Y., Jia, Y., Hill, D.J., Xu, Y., Zhang, Y.: Short-term residential load forecasting based on LSTM recurrent neural network. IEEE Transactions on Smart Grid (2017)
14. Muhammad, A., Lee, J.M., Hong, S.W., Lee, S.J., Lee, E.H.: Deep learning application in power system with a case study on solar irradiation forecasting. In: 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). pp. 275–279. IEEE (2019)
15. Müller, M., Viernstein, L., Truong, C.N., Eiting, A., Hesse, H.C., Witzmann, R., Jossen, A.: Evaluation of grid-level adaptability for stationary battery energy storage system applications in Europe. Journal of Energy Storage **9**, 1–11 (2017)
16. Obaid, Z.A., Cipcigan, L.M., Abrahim, L., Muhssin, M.T.: Frequency control of future power systems: reviewing and evaluating challenges and new control methods. Journal of Modern Power Systems and Clean Energy **7**(1), 9–25 (2019)
17. Pedersen, M.E.H.: Hyper-parameter optimization. https://github.com/Hvass-Labs/TensorFlow-Tutorials (2018)
18. Petneházi, G.: Recurrent neural networks for time series forecasting. arXiv preprint arXiv:1901.00069 (2019)
19. Pumperla, M.: Hyperas: A very simple convenience wrapper around Hyperopt for fast prototyping with keras models. (2017–), http://maxpumperla.com/hyperas/, [Online; accessed 30-04-2019]
20. Ratsimbazafy, M.: McKinsey SmartCities traffic prediction. https://github.com/mratsim/McKinsey-SmartCities-Traffic-Prediction (2018)
21. RTE: Continental Europe synchronous area frequency data metered by RTE. (2019 –), https://clients.rte-france.com, [Online; accessed 30-04-2019]
22. Shi, H., Xu, M., Li, R.: Deep learning for household load forecastinga novel pooling deep RNN. IEEE Transactions on Smart Grid **9**(5), 5271–5280 (2018)
23. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)
24. Toubeau, J.F., Bottieau, J., Vallée, F., De Grève, Z.: Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. IEEE Transactions on Power Systems **34**(2), 1203–1215 (2019)