BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 13/15 pp512–530
DOI: 10.26599/BDMA.2023.9020028
Volume 7, Number 2, June 2024

Improved Quantile Convolutional and Recurrent Neural Networks for Electric Vehicle Battery Temperature Prediction

Andreas M. Billert*, Runyao Yu, Stefan Erschen, Michael Frey, and Frank Gauterin

Abstract: The battery thermal management of electric vehicles can be improved using neural networks predicting quantile sequences of the battery temperature. This work extends a method for the development of Quantile Convolutional and Quantile Recurrent Neural Networks (namely Q*NN). Fleet data of 225 629 drives are clustered and balanced, simulation data from 971 simulations are augmented before they are combined for training and testing. The Q*NN hyperparameters are optimized using an efficient Bayesian optimization, before the Q*NN models are compared with regression and quantile regression models for four horizons. The analysis of point-forecast and quantile-related metrics shows the superior performance of the novel Q*NN models. The median predictions of the best performing model achieve an average RMSE of 0.66°C and R^2 of 0.84. The predicted 0.99 quantile covers 98.87% of the true values in the test data. In conclusion, this work proposes an extended development and comparison of Q*NN models for accurate battery temperature prediction.

Key words: battery temperature; deep learning; convolutional and recurrent neural network; quantile forecasting

1 Introduction

An improvement in energy efficiency of a Battery Electric Vehicle (BEV) reduces energy costs and increases its range. Depending on the share of renewable energies in electricity production, it can also contribute to greenhouse gas emission reduction[1, 2]. Therefore, a Battery Thermal Management System (BTMS) needs to be designed energy efficient due to its impact on vehicle energy consumption[3]. Efficiency can be improved with BTMS components optimized for battery cooling or preheating[4, 5], as well as with an improved BTMS control. For example, Refs. [6–8] show the potential of a predictive BTMS to reduce energy consumption, while taking additional effects, such as battery ageing and derating, into account. Furthermore, data-driven methods can be used for improved battery state estimation, as in Ref. [9].

Predictive control of BTMS uses physical or data-driven models to predict future battery states[6–8, 10–13]. The prediction model needs to provide predictions for sufficiently large horizons while keeping the computational costs low[14, 15]. Physical or empirical battery simulation models can be used for a prediction of the battery thermal behavior, validated with real drive cycle data[16]. Model predictive control of BTMS with neural networks as prediction models shows good performance in real-time control, given their ability to represent complex systems[17–19]. A direct consideration of foresight input data can improve the control strategies, for example using a predicted velocity profile or weather data[8, 20, 21]. Velocity

• Andreas M. Billert is with the Karlsruhe Institute of Technology (KIT), Institute of Vehicle System Technology, Karlsruhe 76131, Germany, and also with Bayerische Motoren Werke (BMW) AG, Munich 80788, Germany. E-mail: andreas.billert@partner.kit.edu.
• Runyao Yu is with the Technical University of Munich (TUM), Munich 80333, Germany. E-mail: runyao.yu@tum.de.
• Stefan Erschen is with BMW AG, Munich 80788, Germany. E-mail: stefan.erschen@bmw.de.
• Michael Frey and Frank Gauterin are with KIT, Institute of Vehicle System Technology, Karlsruhe 76131, Germany. E-mail: michael.frey@kit.edu; frank.gauterin@kit.edu.
* To whom correspondence should be addressed.
Manuscript received: 2023-04-14; revised: 2023-09-16; accepted: 2023-10-09

prediction can be implemented by time-series forecasting using Long Short-Term Memory (LSTM) models[22]. Predictive control can additionally benefit from models that provide the prediction uncertainty, considering the complexity of the physical processes of the BTMS and the uncertainty of the route ahead[7, 23]. The control strategy of a predictive control is derived by optimization costs based on predictions for a moving horizon[11, 13]. For BTMS, cooling energy consumption as well as temperature-dependent battery ageing and power derating can be considered by the cost function[6]. For example, high battery temperatures lead to accelerated ageing and a limitation of maximum available power.

The predictive BTMS proposed in Ref. [6] incorporates a Quantile Convolutional Neural Network (QCNN) for battery temperature prediction. The novel approach reduces battery cooling costs by 9% with unchanged ageing costs. The development of the QCNN is presented in Ref. [24]. It provides the model uncertainty as quantiles of the predicted battery temperature and uses data from the previous 5 km as well as foresight input data of the following 20 km for its predictions. Due to the needed prediction accuracy for cost calculation of the proposed predictive BTMS, an optimization of the prediction model is suggested in Ref. [6].

In Ref. [24], the best performing QCNN shows good performance and its predictions plausibly adapt to the battery cooling thresholds. The predicted quantiles successfully provide information about the prediction uncertainty and do not suffer from quantile crossing. However, both the considered metrics and the exemplary analysis indicate limitiations in the prediction accuracy of the best performing QCNN. Several aspects for performance improvement are discussed in Ref. [24], such as an increased size of datasets as well as improved model architectures and hyperparameters using optimization algorithms. The model should be compared with classic regression models for reference and tested in the predictive BTMS. In the following, these aspects will be introduced in more detail.

The performance of Machine Learning (ML) models depends on the size of the dataset[25–28]. On the one hand, larger datasets with more variance provide more information about the underlying physical processes, cover more scenarios, and allow more complex architectures. On the other hand, large datasets bring additional challenges concerning the handling of the data, such as computation time, anomalies, and redundancy.

Anomaly or outlier detection refers to the identification of rare non-conform data, that can be removed for better performance and robustness of ML models[29, 30–32]. Outlier detection methods differ in complexity, accuracy, suitability for high-dimensional data, and assumptions concerning the occurrence of outliers[31]. Isolation forest from Liu et al.[33] shows to be a suitable method considering its good accuracy and linear complexity[31]. It can be further improved for high-dimensional data by using attribute reduction methods[33]. Dimension reduction methods can address the noise from less relevant features that hide relevant outliers[31]. They can also be used for anomaly detection of time-series data[34]. Uniform Manifold Approximation and Projection (UMAP) from McInnes et al.[35] is a well-performing method for dimension reduction, outperforming Principal Component Analysis (PCA) for numerical data in Ref. [36].

Data balancing is a popular method to reduce data redundancy and improve predictions of rare cases in classification. It can be addressed by data resampling, such as undersampling of classes with high occurrence, oversampling of classes with low occurrence, or combining both[37–39]. Popular methods are random undersampling, random oversampling, and Synthetic Minority Oversampling Technique (namely SMOTE)[40]. Data balancing is also possible for time-series forecasting, for instance based on the distribution of input or output variables, as proposed by Ref. [41] for improved motor temperature predictions of rare cases, and also done in Ref. [24] for battery temperature prediction. In Ref. [41], the best prediction performance is achieved with different target distributions, dependent on the chosen ML model. Alternatively, data balancing for time-series forecasting can be handled by an adapted Deep Learning (DL) architecture as done in Ref. [42]. Data resampling using a relevance bias improves prediction accuracy especially for rare cases of a time-series forecasting model[25, 43]. Resampling of time-series is also possible by distinction of normal and extreme blocks, which increases prediction accuracy for normal events as well as prediction and classification accuracy of extreme events[44]. Random undersampling improves the performance of a Deep Neural Network (DNN) for time-series anomaly detection in Ref. [28].

In Ref. [45], time-series clustering is conducted to divide time-series of battery cell voltage during discharge into a majority and a minority classes. The classes are then resampled to address their imbalance, followed by the training of a Convolutional Neural Network (CNN) for classification. Time-series classification is improved in Ref. [46] by combining data balancing with data augmentation using Fourier transform to iteratively produce surrogates as proposed by Ref. [47].

Data augmentation is a technique that can be applied on time-series data for increasing dataset size and variance in order to reduce model overfitting or to balance the dataset[48, 49]. Augmented data can be obtained in different ways, such as transformation in time or frequency domain (e.g., scaling, stretching, and warping), matching time-series patterns (e.g., SMOTE[40] and Dynamic time warping based Barycentric Averaging (DBA)[50]), or using generative models (e.g., auto-encoder and Generative Adversarial Networks (GAN)[51, 52]). Many methods address data augmentation for time-series classification to increase the accuracy[48, 49, 52–54]. Data augmentation can also improve the performance of time-series regression or forecasting models[49, 51, 55]. Existing data augmentation methods might ignore non-linear correlations and underlying physical dependencies[46] and add noise[52]. Data augmentation using pattern matching can improve model generalization, but need high computation times for long time-series patterns[48].

Various ML methods can be used for time-series prediction, including classic regression methods, as well as DL methods, such as CNN and Recurrent Neural Networks (RNN) (e.g., Gated Recurrent Unit (GRU), LSTM)[56–61]. They can differ in sampling and horizon, they can be uni- or multi-variate and be implemented with a recursive strategy (with one-step prediction) or direct strategy (multi-step prediction)[58, 60, 62]. One-step forecasting can be done by regression models, as implemented for load- and energy forecasting in Refs. [61, 63, 64], using the Python package Pycaret[65] for an automated training and evaluation of different algorithms. Multi-step prediction models, such as CNN, improve the prediction performance[66], while recursive strategies can be subject to an increasing error over the prediction horizon[60]. An example of battery temperature prediction with LSTM is provided by Ref. [67] using real-world driving data, with a focus on seasonal characteristics and thermal faults. In Ref. [68], LSTM is further applied to a synchronous prediction of the battery temperature, State Of Charge (SOC), and voltage. Additionally, LSTM improves the State Of Health (SOH) estimation in Ref. [69] and in combination with CNN in Ref. [70]. A combination of CNN and LSTM obtains accurate SOC estimation in Ref. [71] and remains useful life estimation in Ref. [72] with an auto-encoder for feature augmentation.

Model uncertainty can be provided as quantile predictions for a predictive BTMS[6]. Quantile regression can be used for time-series forecasting, such as in Quantile Random Forest (QRF) and Quantile Extra Trees Regressor (QETR) from the Python package scikit-garden[73]. It can be combined with neural networks[74, 75], CNN[24, 76, 77], RNN[76–78], and other DL models[77, 79]. The models from scikit-garden can be used as reference models, as done by Refs. [75, 78, 79]. For example, different one-step prediction models can be trained for different horizons[78].

Optimization algorithms can be used to tune hyperparameters of ML models to increase their performance[80]. A popular method is Bayesian optimization, which tunes hyperparameters with consideration of results from previous trials[81, 82]. It can find better hyperparameters in shorter computation time than grid search or random search[80]. The computation time of Bayesian optimization can be further improved by parallelized optimization on subsets of the training data[83] or by using training data subsets with increasing size, until the whole dataset is used in the last stage[84]. Improved performance is also achieved by the hyperband method[85] and a combination of hyperband and Bayesian optimization[86]. Besides model hyperparameters, parameters of data balancing methods can be included in the search space as well[87, 88]. Bayesian optimization has been successfully used for hyperparameter optimization of time-series forecasting models in Refs. [89, 90].

In this work, improved battery temperature prediction models are developed. The models are based on the method of developing QCNN from Ref. [24]. They predict quantile sequences of the battery temperature change, including the prediction accuracy. The method is adapted to make use of a larger dataset. The contributions of this work include clustering-based data balancing and a novel data augmentation

technique related to pattern matching. Besides using a CNN architecture, models using GRU and LSTM architectures are developed. The hyperparameters of each architecture are optimized using an efficient Bayesian optimization approach. Regression and quantile regression models are trained on the same dataset for comparison, which concludes the contributions of this work.

The nomenclatures used in this paper are shown in Fig.1.

Latin symbols

A	Segment A of a trip
B	Segment B of another trip
k	Horizon size
L	Loss
n	Number (quantity)
p	Occurrence
q	Quantile
s	Share of training data
T	Temperature
t	Trial
v	Speed
X	Reduced dimension X
Y	Reduced dimension Y

Abbreviations and acronyms

ADAM	ADAptive Moment estimation
BEV	Battery Electric Vehicles
BTMS	Battery Thermal Management System
CNN	Convolutional Neural Network
CORS	Crossover Rate Score
DBA	Dynamic time warping based Barycentric Averaging
DL	Deep Learning
DOE	Design Of Experiments
ETR	Extra Trees Regressor
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
IBA	Intersection-Based Assembly
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
max	Maximum
min	Minimum
ML	Machine Learning
MSE	Mean Squared Error
PCA	Principle Component Analysis
QCNN	Quantile Convolutional Neural Network
QRNN	Quantile Recurrent Neural Network
Q*NN	Quantile Neural Network based on a deep learning architecture (e.g., with convolution or recurrent layers)
QNN	Quantum Neural Network
QETR	Quantile Extra Trees Regressor
QGRU	Quantile Gated Recurrent Unit
QL	Qualifier
QLSTM	Quantile Long Short-Term Memory
QRF	Quantile Random Forest
R^2	Coefficient of determination
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SDT	Shallow Decision Tree
std	Standard deviation
SMOTE	Synthetic Minority Oversampling Technique
SOC	State Of Charge
SOH	State Of Health
UMAP	Uniform Manifold Approximation and Projection
US06	US06 drive cycle
WLTC	Worldwide harmonized Light vehicles Test Cycle
WS	Winkler Score

Greek symbols

α	Weight
Δ	Difference
μ	Quantile weight
ψ	Sharpness

Subscripts

1	Referring to the part of a trip before a segment (A or B)
2	Referring to the part of a trip after a segment (A or B)
a	Ambient
avg	Average
b	Battery
b, cool, lower	Lower battery cooling threshold
b, cool, upper	Upper battery cooling threshold
d, abs	Absolute difference
i	Index i, related to horizon step
j	Index j, related to quantile
μ	Quantile weight
MSE	Mean Squared Error
pb	Pinball

Fig. 1 Nomenclature.

2 Method and Material

The proposed method for improved battery temperature prediction and model comparison is described in this chapter. The main steps are shown in Fig. 2. At first, simulation data and vehicle fleet data are collected. Time-series clustering is applied to the fleet dataset for data understanding, outlier removal, and balancing. The simulation dataset is augmented for better data balance between fleet data and simulation data, before both are used for model building and training. Custom quantile prediction models using different neural network architectures are developed with Bayesian optimization for hyperparameter tuning. Models are retrained for the best hyperparameter sets. Reference models using regression and quantile regression are trained on the same datasets. The reference models provide one-step predictions rather than time-series, such that one model for each of four different horizons is developed. All models are tested, compared, and evaluated with respect to an application for predictive BTMS. Model application (inference) consists of vehicle data collection during a trip, data preprocessing, and battery temperature prediction with the chosen model.

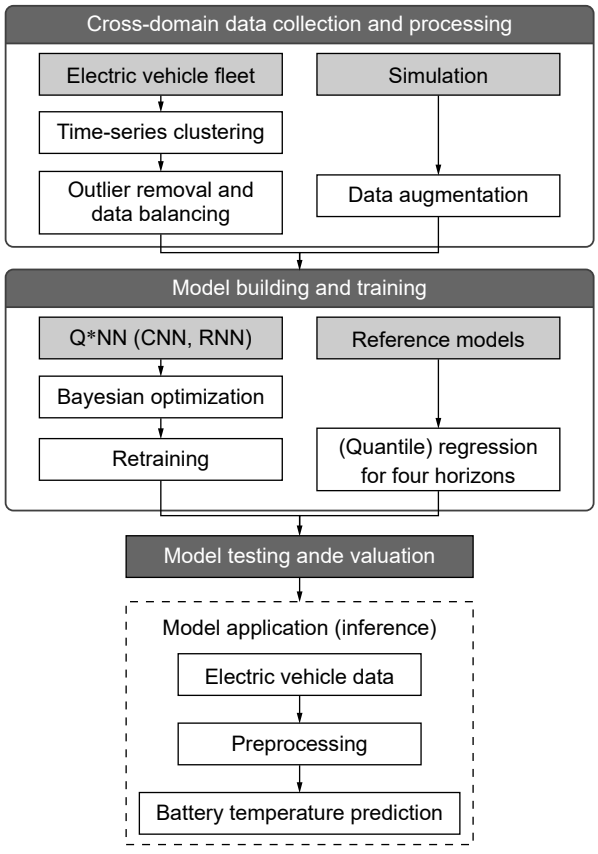

Fig. 2 Pipeline for model development and comparison using cross-domain input data, as well as main steps for application (inference) in a vehicle.

2.1 Cross-domain data collection and processing

This section focuses on the collected datasets and the processing techniques, such that they can be used for model building and training.

2.1.1 Vehicle fleet data

Compared to Ref. [24], a new data source is used which consists of a larger amount of fleet data. In total, the available fleet dataset covers 225 629 drives with 9 917 426 km total drive distance between December 2021 and October 2022. In contrast to Ref. [24], additional weather data cannot be joined to the new vehicle fleet dataset because it does not include location data. Only trips with distances larger than 20 km and battery temperatures higher than 15°C are considered, since the focus of this work is on the cooling behavior. The battery temperature is clustered as sequences using the mini-batch k-means algorithm from the Python package scikit-learn for a window size of 20 km and an overlap of 25%. The number of clusters is determined using elbow method (with sum of squared distance, the lower the better) and silhouette score (the higher the better)[91]. UMAP is used for dimension reduction to two dimensions in order to visualize the clustering result. Outliers are removed based on the reduced dimensions using isolation forest with a threshold of 1%. The remaining time-series data are balanced with random undersampling to a maximum of 20 000 time-series per cluster. After the data are split into training (70%), validation (15%), and test data (15%), random oversampling is applied to each cluster in each of the datasets. The results of clustering and balancing are provided in Section 3.1.1.

2.1.2 Simulation data

Simulation data are based on the same Design Of Experiments (DOE) than in Ref. [24], but with more simulated profiles. It consists of 971 simulations with a drive time of one hour, covering five battery cooling thresholds (25°C, 30°C, 35°C, 40°C, and 45°C). The included simulation test dataset is the same as that in Ref. [24]. Due to the increased vehicle fleet dataset, the share of simulation data is significantly lower. In order to address this imbalance, the simulation dataset for training and validation is augmented using a time-series data augmentation method proposed in this work. Additionally, a higher overlap of 75% is chosen

for the simulation training and validation dataset, and all test data are used (i.e., with no overlap filter).

The novel data augmentation method proposed in this work is referred to as Intersection-Based Assembly (IBA). Figure 3 shows how an extended dataset can be composed of the original dataset and augmented data. The method searches for segments that match given criteria for each data point (250 m segment) in the original data. A match is found if the difference between the signal values is within a matching tolerance. The tolerance can be adjusted dependent on the amount of data needed and the tolerated value difference. As additional constraint, the drive of the matched segment has to be from another velocity profile and has the same battery cooling threshold. Furthermore, cabin air-conditioning has to be in the same state (on or off) and the segment may not be from the first 20 segments of the matched drive. For each match, possible matches with neighboring segments are filtered (with a tolerance of 8 segments). The augmented data are extracted by combining the part of the drive before Segment A with the part after the matched Segment B (i.e., a combination A_1B_2) and vice versa (B_2A_1). Both new drives are cut, such that the intersection of the two drives is included in either the history or foresight horizon of the model. This avoids the duplication of existing data points from the original dataset. An example for augmented data is provided in the results section.

2.2 Model building and training

The collected and processed datasets are combined for model building and training. Vehicle fleet data and simulation data are mixed and treated equally during training. The same data are used for the development of the proposed Quantile Neural Network DL architecture (namely Q*NN), as well as for the reference models.

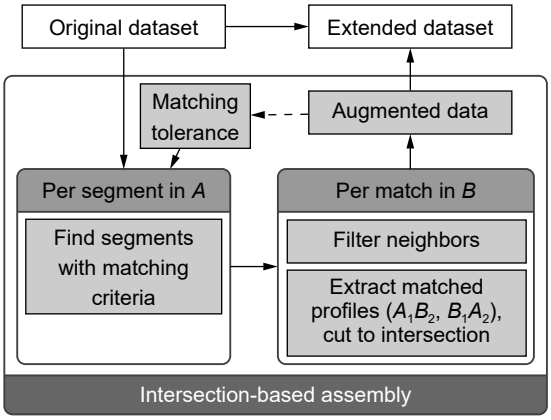

Fig. 3 Time-series data augmentation using IBA. For each segment in a (simulated) drive *A*, the method looks for a segment in another drive *B* that matches predefined criteria. Augmented data are composed by the drives around the matched segments.

2.2.1 Q*NN

In this work, the proposed architecture is called Q*NN (not related to Quantum Neural Networks (QNN)[92]). It combines quantile sequence predictions with one-dimensional CNNs or with one of two types of RNNs, namely GRU and LSTM. The models are referred to as QCNN, QGRU, and QLSTM. The base architecture is the same as that in Ref. [24], which includes an architecture diagram. In the following, only the key elements and the differences to Ref. [24] are further described. The architecture consists of two input channels (history and foresight), separately processing input data of the previous 5 km and the following 20 km. A list of the input features is provided in Table A1 in Appendix A. The sequential information are processed by blocks with 1D convolutional (QCNN) or recurrent (QGRU and QLSTM) layers. After separate processing of history and foresight input data, they are flattened and merged in a combined channel with additional convolutional or recurrent layers. Seven output layers provide a quantile sequence prediction of the battery temperature change with respect to the current temperature. In case of the RNN models with recurrent layers, the MaxPooling layers are removed. Dropout, batch normalization, data balancing, and data augmentation are applied to reduce overfitting. The layer parameters are the same as those in Ref. [24]. The used activation functions are “relu” and “linear” (for each quantile output layer). ADAptive Moment estimation (ADAM) optimizes the total loss during training with an early stop patience of 20. The custom layer from Ref. [24] is used for loss calculation L_{total} of k segments in the prediction horizon, including n_μ constrained quantile weights μ,

$$L_{\text{total}} = \frac{1-\alpha_{\text{MSE}}}{k\,(2n_\mu+1)} \sum_{j=0}^{2n_\mu} \sum_{i=1}^{k} \mu_j L_{\text{pb}}\,(i,j) + \alpha_{\text{MSE}} \times L_{\text{MSE}} \quad (1)$$

where L_{total} is a weighted sum of the Mean Squared Error (MSE, as L_{MSE} with weight α_{MSE}) and pinball loss (L_{pb}) from quantile regression. It considers over- and under-estimation differently dependent on the corresponding quantile[77].

The hyperparameters are tuned by Bayesian Optimization. Their ranges are defined as follows (and based on Ref. [24]):

• Base number of layer nodes (filters for 1D convolutional layers and units for GRU or LSTM layers) in range [128, 256] with step size 32;

• Dropout rate in range [0.2, 0.4];

• Loss weight α_{MSE} in range [0.01, 0.1];

• Learning rate in range [0.001, 0.000 01] with logarithmic sampling;

• Mini-batch size in range [32,128] with step size 16;

• Number of convolutional or recurrent layers in the combined channel (after merge) either two or four (the latter is referred to as the deeper network).

The TensorFlow/Keras package is used for Q*NN building and training in Python. The hyperparameters are optimized using the Keras Tuner implementation of Bayesian optimization with 20 trials (steps) and 3 (QLSTM: 2) executions (models) per trial. For each execution, a model is trained from scratch with the same hyperparameters, and the lowest validation loss of all training epochs is taken. The validation loss of the trial is calculated as average of the executions, which is used by the Bayesian optimization algorithm to evaluate the hyperparameters of the trial. The usage of several executions per trial addresses the random initialization of neural network weights which can lead to bad training even for good hyperparameters. In this work, Bayesian optimization is used for all three neural network architectures (QCNN, QGRU, and QLSTM). In order to reduce computation time, an efficient Bayesian optimization approach is proposed, similar to Ref. [84]. The amount of training data is reduced for the first trials (by random selection), with increasing training data size for each trial. In this work, the share of training data per trial s is defined as follows, with trial number t and the total number of trials n_t:

$$s(t)=\frac{\left\lceil 100\times\left(0.1+\frac{0.9}{1+\exp\left(-12\times\left(\frac{t}{n_t}-0.4\right)\right)}\right)\right\rceil}{100} \tag{2}$$

The random training data selection is performed individually for each execution per trial, which addresses the impact of a potentially bad data selection on model training. After the Bayesian optimization evaluated all trials, the hyperparameters of the trial with lowest validation loss are selected. Three models are trained with the according hyperparameters using the whole training dataset. The best performing model will be used in the next steps of model testing and comparison.

2.2.2 Reference models

In this work, reference models are created for a better understanding of the Q*NN performance. For each prediction horizon of 5 km, 10 km, 15 km, and 20 km, the prediction task is simplified to a single-value regression. The prediction output is the difference between current battery temperature and the temperature at the end of the according horizon. The input foresight features are reshaped to the following 12 input values calculated for each horizon:

• Battery temperature: current value;

• Ambient temperature: current value;

• Upper battery cooling threshold: current value;

• Lower battery cooling threshold: current value;

• Velocity: maximum, minimum, mean, and standard deviation.

• Velocity difference: maximum, minimum, mean, and standard deviation.

Based on the reshaped data, regression models can be trained for each horizon. The Python package Pycaret[65] is used for training and comparison of classic regression models. The best model according to the Root Mean Squared Error (RMSE) is chosen for each horizon. A Shallow Decision Tree (SDT) is trained as a reference model with low complexity, which has lower computational requirements. This allows an investigation if such a simple rule-based model is sufficient. Furthermore, the SDT serves as additional reference for the prediction performance of more complex models (e.g., Q*NN). Hyperparameter optimization for the SDT is conducted using grid search to find the best suitable maximum tree depth and number of leaves. Scikit-garden[73] is used for the training of a quantile regression model, which provides single-step quantile predictions. In total, twelve models are used for testing and comparison with the Q*NN models, consisting of a regression model, a shallow regression model, and a quantile regression model for each of the four horizons.

2.3 Model testing, evaluation, and application

The developed models are evaluated with point-forecast and quantile-related metrics for the test dataset as described in Ref. [24]. Point-forecast metrics treat a predicted quantile as a classic time-series forecast. The focus of the point-forecast metrics is on the 0.5-quantile (median) prediction since it is the closest to the true values on average (by definition). It can be

compared with the reference regression models, for which the predicted values for the four reference horizons are taken. The smoothness is the only metric that cannot be compared, because it requires sequences which are not provided by the reference models. Quantile-related metrics describe quantile properties, including quantile crossing, the width of the quantile intervals, and the share of true values within the predicted quantiles compared with their definition, also known as calibration[93]. They are additionally calculated for the four horizons of the quantile regression models. Further comparison includes the different Q*NN architectures and the best performing QCNN from Ref. [24] (in this work referred to as QCNN22). Besides the calculated metrics, further evaluation includes regression plots and exemplary analysis.

3 Result

The results of the proposed development and comparison of battery temperature prediction models are presented in this section. Its structure mirrors Section 2.

3.1 Cross-domain data collection and processing

The presented methods for time-series data balancing and augmentation are used to provide a balanced dataset for model training and evaluation. They are applied on the collected vehicle fleet data (balancing) and simulation data (augmentation).

3.1.1 Vehicle fleet data

Time-series clustering is applied on the change in battery temperature as sequence over a horizon of 20 km. 7 clusters are obtained using mini batch k-means with elbow method and silhouette score. The time-series data are reduced to two dimensions using UMAP, and visualized in Fig. 4 with the according cluster number. The clusters show only few overlap which indicates a successful clustering. After isolation forest has been applied with 1% outlier removal, the DBA of each cluster is calculated and presented in Fig. 5. The data share of each cluster is included next to each DBA. Data without a change in battery temperature (Cluster 4) takes a share of 53% of the whole fleet dataset. 31% of the data show a rising battery temperature (Clusters 1 to 3) and 16% of the data show a falling battery temperature (Clusters 5 to 7). For all DBA of rising or falling temperature, bigger

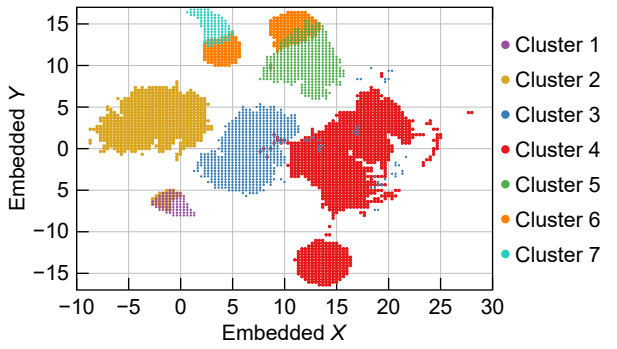

Fig. 4 Reduced dimensions X and Y of the change in battery temperature using UMAP, with labels from time-series clustering.

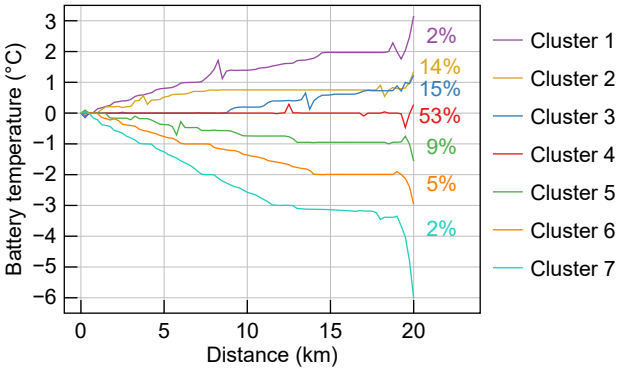

Fig. 5 DBA of the battery temperature change for each cluster after outlier removal. The share within the fleet dataset is included next to each cluster.

changes plausibly occur at later points in the horizon. Data balancing is needed considering the difference in the shares between the clusters, because a good prediction accuracy is also required for higher changes in battery temperature.

3.1.2 Simulation data

The proposed data augmentation method IBA is applied on the simulation dataset to increase its share compared to the vehicle fleet data size. The amount of augmented data depends on the chosen matching tolerance. Figure 6 shows this dependency for the given simulation dataset. The augmentation factor describes the increase of the total dataset. The matching tolerance is chosen to the smallest value that results in the required data size. For this work, a tolerance of 3% is chosen for a 1.64 times larger simulation dataset, such that the amount of simulation data roughly equals the amount of balanced fleet data per cluster. An example of augmented profiles is provided in Figure A1 in Appendix B.

The data used in the following steps consist of the

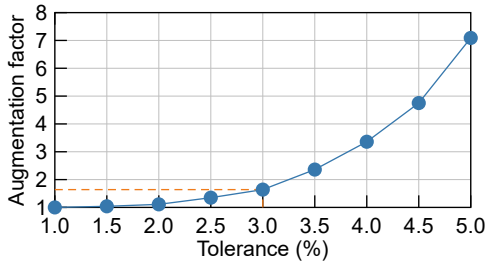

Fig. 6 Achieved augmentation factor of IBA dependent on the tolerance, for the given simulation dataset. In this work, a tolerance of 3 is chosen (dashed line).

balanced vehicle fleet dataset and the augmented simulation dataset. Table 1 shows the dataset sizes.

3.2 Model building and training

The results from model training of both the proposed Q*NN and the reference models are presented in this section. The best performing model is selected for each model type based on the validation loss. The hardware used for model training and testing consists of an Intel Xeon Silver 4114 (2.2 GHz), 64 GB RAM, and a NVIDIA Quadro RTX 4000. The TensorFlow/Keras package version is 2.5.0 and the Python version is 3.7.5.

3.2.1 Q*NN

The hyperparameters of each Q*NN architecture are obtained using the proposed efficient Bayesian optimization. The average validation loss of each trial and each architecture is shown in Fig. 7, with the share of training data as gray shaded area in the background. The trials with lowest validation loss are trial 18 (QCNN), trial 20 (QGRU), and trial 11 (QLSTM). The average training time per model is 1.87 h (QCNN), 2.78 h (QGRU), and 3.51 h (QLSTM). For each architecture, three models are trained with the best hyperparameters, and the model with lowest validation loss is used in the following comparisons. Table A2 in Appendix A shows the hyperparameters of the best trials with lowest validation loss.

3.2.2 Reference models

The RMSE is calculated for the validation data set for all 18 reference regression models compared by

Table 1 Share and number of data points (in brackets unique drives) for each dataset after preprocessing.

Dataset	Training	Validation	Test
Vehicle fleet	86.5% (94.8%)	80.4% (92.7%)	85.7% (99.8%)
Simulation	13.5% (5.2%)	19.6% (7.3%)	14.3% (0.2%)
Total	142 809 (52 333)	13 634 (7607)	13 107 (7072)

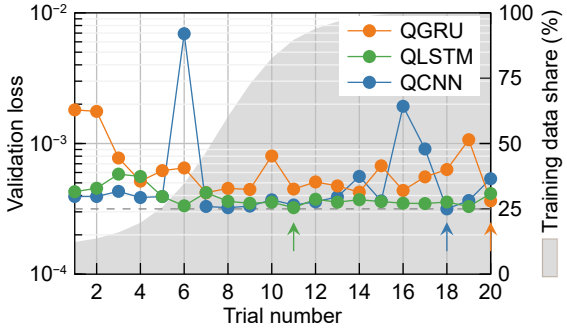

Fig. 7 Validation loss for all trials of efficient Bayesian optimization of all three Q*NN architectures. For each trial, the average of the lowest validation loss of three (QLSTM: two) trained models is shown. The best trial is marked with an arrow.

Pycaret. The results for the four considered horizons are shown in Table 2. For each horizon, the best performing model is chosen as reference model for model testing. This means Light Gradient Boosting Machine (LGBM) is chosen for a horizon of 5 km and Extra Trees Regressor (ETR) for 10 km, 15 km, and 20 km. The reference model is referred to as LGBM-ETR. Each model consists of 100 estimators, i.e., boosting stages for LGBM and trees for ETR. Their feature importance is included in Fig. A2 in Appendix C. For all four horizons, the top three features are the current ambient temperature, battery temperature, and the maximum velocity.

The SDT hyperparameters are optimized by grid search on the following ranges:

- Maximum tree depth: 4, 5, 6, 7, 8, 9;
- Maximum number of leaves: 9, 11, 13, 15, 17.

Both ranges are limited such that the resulting tree will be shallow, thus easier to implement as a rule-based model. The hyperparameters with the lowest validation loss are 7 (tree depth) and 17 (number of leaves) for the 5 km horizon, and 5 and 17 for the other horizons (10 km, 15 km, and 20 km).

The quantile regression model is based on an ETR, since ETR shows the overall best performance in the comparison of regression models by Pycaret. In this work it is referred to as Quantile Extra Trees Regressor (QETR) and trained using scikit-garden. According to the Pycaret models, 100 estimators are chosen. The minimum sample split is set to 0.01. The feature importance of all four resulting models is also included in Fig. A2 in Appendix C. The three most important features are battery temperature, ambient temperature, and the upper battery cooling threshold.

Table 2 RMSE (°C) of each regression model for each reference horizon in the validation data. The table is sorted by the values for 20 km, and the best value per horizon marked in bold. The top three models are separated by a line. All models are trained using Pycaret.

(°C)

Model	5 km	10 km	15 km	20 km
Extra trees regressor	0.58	**0.87**	**1.12**	**1.33**
Light gradient boosting machine	**0.57**	0.88	1.14	1.36
Random forest regressor	0.59	0.89	1.16	1.39
Gradient boosting regressor	0.60	0.92	1.20	1.44
Decision tree regressor	0.77	1.13	1.47	1.78
K neighbors regressor	0.70	1.10	1.47	1.79
Huber regressor	0.66	1.06	1.46	1.80
Bayesian ridge	0.66	1.07	1.46	1.81
Least angle regression	0.72	1.07	1.46	1.81
Linear regression	0.66	1.07	1.46	1.81
Ridge regression	0.66	1.07	1.46	1.81
AdaBoost regressor	0.76	1.28	1.76	2.16
Dummy regressor	0.76	1.28	1.77	2.18
Elastic net	0.76	1.28	1.77	2.18
Lasso least angle regression	0.76	1.28	1.77	2.18
Lasso regression	0.76	1.28	1.77	2.18
Orthogonal matching pursuit	0.77	1.29	1.80	2.19
Passive aggressive regressor	1.42	1.14	1.54	2.93

3.3 Model testing, evaluation, and application

The best performing Q*NN and reference models are compared using their predictions on the test data. The comparison is divided into point-forecast and quantile-related comparison. A prediction example is included for better understanding of the model differences.

3.3.1 Point-forecast comparison

All models can be compared as a one-step prediction for the respective horizon. For the quantile models (Q*NN and QETR), the 0.5-quantile (median) prediction is used for the comparison. Figure 8 depicts the RMSE and coefficient of determination (R^2) metrics for the four horizons. A smaller RMSE and a higher R^2 value indicate better performance. A table of the metrics averaged over the horizons is included in Table A3 in Appendix D.

The Q*NN models show the best performance, followed by the LGBM-ETR model from Pycaret. QGRU performs better than QCNN and QLSTM, which are on a similar level. The QGRU achieves an RMSE of 0.66°C and an R^2 of 0.84. The absolute difference between its median prediction and the true values is maximum 1.00°C in 90% of the test data,

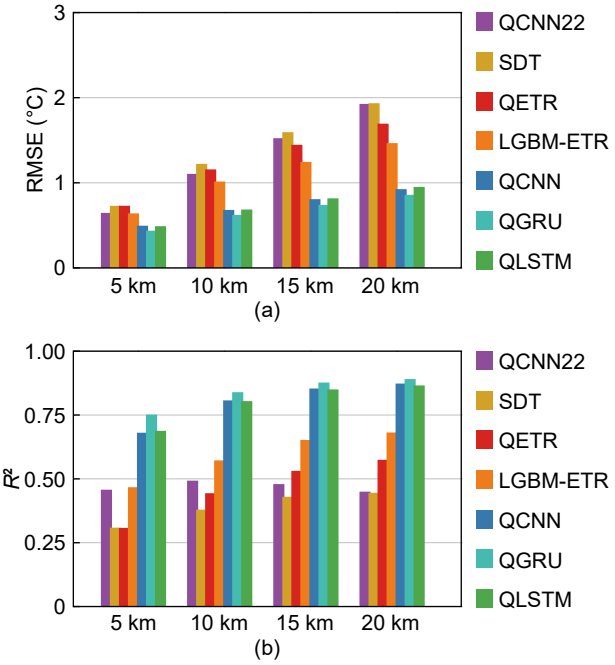

Fig. 8 Point-forecast metrics (a) RMSE and (b) R^2 of all models on the test data for four horizons.

averaged over the four horizons. QGRU will be used as best QRNN model for the following comparisons. The LGBM-ETR model is followed by QETR and QCNN22 (on rank five and six). Their order depends on the horizon, for which QCNN22 performs better for 5 km and 10 km, and QETR performs better for 15 km and 20 km. The SDT performs worst for all horizons.

Regression plots for the best Q*NN (Figs. 9a and 9b), QCNN22 (Fig. 9c), and reference models (Figs. 9d and 9f) are provided in Fig. 9. The difference between true value and prediction is marked in different colors for all four horizons. The prediction models plausibly predict larger battery temperature changes for larger horizons. Better performance is achieved when the shape of the point cloud follows the identity line (dashed). Thus, LGBM-ETR (Fig. 9d) shows worse performance compared to the novel Q*NN models (Figs. 9a and 9b), particularly for larger changes in battery temperature. The Q*NN models slightly differ in performance for increasing battery temperature. For instance, the QGRU model (Fig. 9b) shows more overestimation but less underestimation of the true change compared with the QCNN model (Fig. 9a) for horizon 15 km and 20 km. QCNN22 (Fig. 9c) largely underestimates the battery temperature change in a few cases, but except that follows the identity closer than LGBM-ETR (Fig. 9d). The SDT (Fig. 9e) shows worst

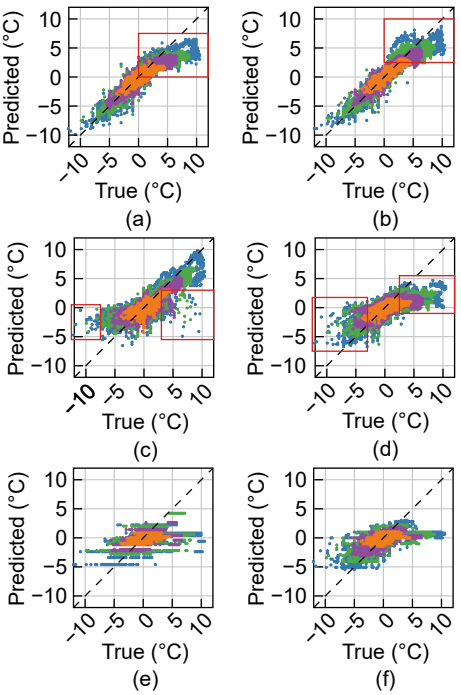

Fig. 9 Regression plots of the battery temperature predictions for the test data. The plots are shown for (a) QCNN, (b) QGRU, (c) QCNN22, (d) LGBM-ETR, (e) SDT, and (f) QETR. The colors indicate the horizon, increasing from inner to outer with 5 km (orange), 10 km (purple), 15 km (green), and 20 km (blue). The rectangles (red) mark areas discussed in the text.

performance with an almost horizontal point cloud shape, which means it underestimates larger changes in battery temperature. The performance of the QETR model (Fig. 9f) is between the LGBM-ETR and SDT, with a significant underestimation of high rises of battery temperature by its median prediction.

An error histogram of the best performing model QGRU is given in Fig. 10. In 91.89% of the test data, the error is between −1 and 1°C, and in 98.38% between −2 and 2°C. The peak of the histogram is located at 0.2°C, which indicates a small overestimation of the battery temperature change.

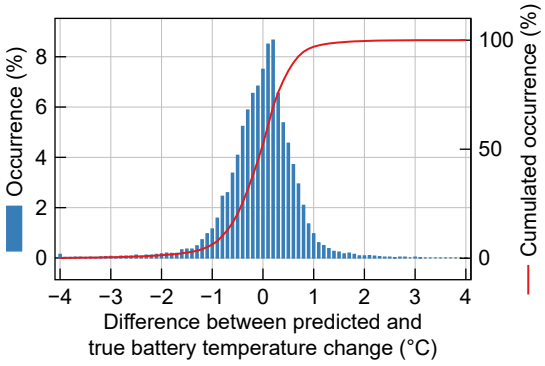

Fig. 10 Error histogram of the QGRU with steps of 0.1°C.

3.3.2 Quantile-related comparison

One-step quantile predictions are obtained from the quantile models (Q*NN, QCNN22, and QETR) for all four horizons. The quantile-related metrics from Ref. [24] are calculated, and their average over the horizons is shown in Table 3. Better performance is achieved with lower values for all metrics. Additionally, the share of true values below the 0.5 quantile (median) and 0.99 quantile is given as average over the horizons.

The QCNN22 model shows the best performance for the average sharpness ψ_{avg}, but performs worst for the occurrences within quantiles $p\,(q_j)_{\text{d, abs}}$, and quantile intervals $p\,(q_j, 1-q_j)_{\text{d, abs}}$, as well as the Winkler Score (WS). Except for quantile crossing indicated by Crossover Rate Score (CORS), the Q*NN models from this work perform better than QETR, but no model outperforms in all metrics. For the QGRU predictions, the true values are lower than the median prediction in 56.19% of the test data. The true values for the 0.99 quantile prediction are lower in 98.87% for the QGRU. The reference model QETR performs comparably well for the 0.99 quantile prediction (98.29%), but bad for the median (40.40%).

Table 3 Quantile-related metrics on the test data, average over all four horizens. Bold values indicate the best value for each metric.

Model	Metric							
	$p\,(q_j)_{\text{d,abs}}$	$p\,(q_j, 1-q_j)_{\text{d,abs}}$	WS	ψ_{avg} (°C)	ψ_{max} (°C)	CORS ($\times 10^{-3}$)	0.50 quantile (%)	0.99 quantile (%)
QETR	0.05	0.06	1.54	2.67	7.67	**0**	40.40	98.29
QCNN22	0.20	0.44	16.81	**1.03**	6.10	1.83	41.27	68.95
QCNN	**0.03**	0.05	2.42	1.37	4.51	12.09	**46.35**	95.99
QGRU	0.04	0.06	**1.27**	1.54	**4.11**	0.02	56.19	**98.87**
QLSTM	0.05	**0.02**	1.39	1.86	5.37	1.70	60.10	98.13

3.3.3 Prediction example

Further comparison can be conducted by exemplary analysis of the predictions on test data. Figure 11 depicts the prediction on a test fleet data sample for the QCNN (Fig. 11a), QGRU (Fig. 11b), QCNN22 (Fig. 11c), and one-step reference models (Fig. 11d). The QGRU model follows the true values closer than the QCNN model, as can be seen in the distance 0 km to 5 km (with negative battery temperature change) and 15 km to 20 km where the median is closer to the true values. This is one example that shows the underestimation of the QCNN compared with the QGRU as observed by the regression plots. Furthermore, the range of the predicted quantile intervals is smaller for the QGRU. For instance, the QGRU model predicts a change in battery temperature between 3°C and 6°C after 20 km with 98% certainty, while the QCNN predicts a range of 1.5°C to 6.5°C.

In comparison with the Q*NN models, predictions of the reference models provide less information. While each Q*NN model provides quantiles as sequences over the prediction horizon, the reference models only provide a single-step prediction for one horizon, without a connection between the different steps in the horizon. In the shown example, the LGBM-ETR model shows comparable results to the median prediction of the QCNN. This is followed by the median predictions of the QETR and the prediction of the SDT with the largest underestimation of the battery temperature change. The quantile intervals of QETR are much wider than for the Q*NN predictions. Additionally, the true values are more often in the outer quantile intervals with lower predicted model certainty of the QETR.

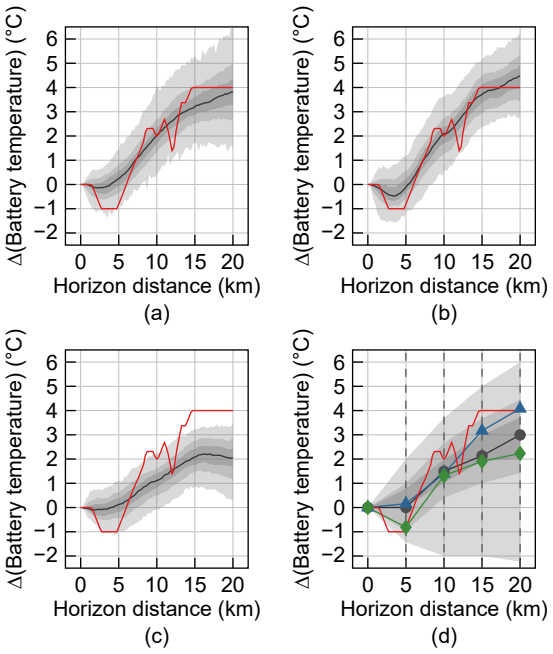

Fig. 11 Predictions for a test drive starting at 15°C battery temperature of the models (a) QCNN, (b) QGRU, (c) QCNN22, and (d) the reference models. The latter are QETR (quantiles and circles), LGBM-ETR (triangles) and SDT (diamonds). As in Ref. [24], the predicted quantiles are represented by shaded areas for the intervals (0.01/0.99, 0.1/0.9, and 0.25/0.75 from outer to inner), and boundaries from bottom (0.01) to top (0.99). The median is a thin line (black), the true values a thick line (red). In (d), vertical dashed lines indicate the prediction horizons for the single-step reference models.

4 Discussion

This work addresses several aspects discussed in Ref. [24] for battery temperature prediction, such as the need for larger datasets and their handling. Additionally, the dependency of horizon on prediction performance has been considered. Model building includes recurrent layers (GRU and LSTM) and deeper networks. Bayesian optimization is used instead of random grid search for more efficient hyperparameter optimization. The resulting models are compared with regression and quantile regression models for further evaluation.

As a consequence of the addressed aspects, the novel Q*NN models perform better than the QCNN22 model from Ref. [24]. The QCNN22 model has not been trained on the larger fleet data. Consequently, it cannot predict well new situations presented in the new fleet data due to possible overfitting. The new models are trained on the bigger fleet data with clustering-based data balancing that is assumed to cover more scenarios. Thus, they are expected to perform better in later application. However, there remain aspects need to be further investigated.

The proposed clustering of the battery temperature enables an efficient analysis and data balancing. Novel data augmentation for simulation data allows an increase in training data by assembling existing data while preserving physical dependencies. However, the effect of both parts of data preparation has not been examined independently. This work does not provide quantified information about the individual effect of

these methods on the resulting prediction performance. While this work compared convolutional with recurrent Q*NN architectures, following works can combine both layer types into one architecture as has been investigated in other applications[94]. The efficient Bayesian optimization allows faster hyperparameter optimization using an increasing share of training data over the trials. However, future works need to investigate the effect on reduction of optimization time, and compare the prediction performance of models trained with the resulting hyperparameter sets.

Further evaluation of the prediction model performance can focus on model application in a predictive control. The prediction models can be tested and compared in the predictive BTMS as introduced in Ref. [6]. Such an analysis provides information about the needed complexity of prediction models for a predictive BTMS.

5 Conclusion

Different machine learning models for battery temperature prediction of battery electric vehicles are trained and compared in this work. The following contributions are achieved:

• Data processing included time-series clustering, outlier removal, and data balancing of a large fleet dataset of 225 629 drives.

• The simulation dataset of 971 simulations is augmented using the newly proposed method IBA.

• QCNN and recurrent neural networks (QGRU and QLSTM) are trained for battery temperature prediction as quantile sequences, with an efficient Bayesian hyperparameter optimization.

• The models are compared with the model from Ref. [24], one-step regression models, and quantile regression models for four different horizons.

• The best performing model QGRU achieves an average RMSE of 0.66°C and an R^2 of 0.84, which is a large improvement compared to 1.30°C (RMSE) and 0.47 (R^2) for the model from Ref. [24].

• The QGRU predictions differ from the true values by maximum 1.00°C in 90% of the test data. The 0.99 quantile prediction covers 98.87% of the true values.

The results show accurate battery temperature prediction of the developed Q*NN including the prediction uncertainty. In a future work, the models can be integrated in a predictive Battery Thermal Management System (BTMS), such as Ref. [6] for increased efficiency and performance comparison in application. The proposed methods should be further investigated considering their individual impact and applied to other domains.

Appendix

Table A1 Features of the history input and foresight input channel of the Q*NN.

Parameter	History	Foresight
Acceleration pedal (%)	1	0
Ambient temperature (°C)	1	1
Battery cooling threshold end (°C)	1	1
Battery cooling threshold start (°C)	1	1
Battery SOC rate (%/km)	1	0
Battery SOC squared (%^{2})	1	0
SOC (%)	1	0
Battery temperature (°C)	1	0
Battery temperature-ambient temperature (°C)	1	0
Battery temperature rate (°C/km)	1	0
Cabin target temperature (°C)	1	0
Cabin target temperature QL	1	0
Cabin temperature (°C)	1	0
Cabin temperature QL	1	0
DPMA acceleration propulsion (m/s^2)	1	0
DPMA acceleration propulsion QL	1	0
DPMA acceleration recuperation (m/s^2)	1	0
DPMA acceleration recuperation QL	1	0
DPMA speed propulsion (km/h)	1	0
DPMA speed propulsion QL	1	0
DPMA speed recuperation (km/h)	1	0
DPMA speed recuperation QL	1	0
Electric machine stator temperature (°C)	1	0
Electric machine stator temperature QL	1	0
Inverter temperature (°C)	1	0
Inverter temperature QL	1	0
Road height difference negative sum (m/km)	1	1
Road height difference negative sum QL	1	1
Road height difference positive sum (m/km)	1	1
Road height difference positive sum QL	1	1
Speed (km/h)	1	1
Speed difference (km/h)	1	1
Speed difference squared ((km/h)^2)	1	1
Speed inverted (h/km)	1	1
Speed to the power of five ((km/h)^5)	1	1
Temperature after heat pump (°C)	1	0
Temperature after heat pump QL	1	0
Vehicle torque (Nm)	1	0
Within horizon QL	1	1

A

All history inputs and foresight inputs of the Q*NN models are listed in Table A1 as in Ref. [24]. Qualifier Values (QL) are included if needed for the corresponding feature. If a feature is used in the history input channel or foresight input channel, it is marked with 1 in the corresponding column. The Q*NN hyperparameters of the trials with lowest validation loss are shown in Table A2.

B

Figure A1 shows an example of augmented profiles. The example profiles are composed of the Worldwide harmonized Light vehicles Test Cycle (WLTC)[95] and the US06 cycle[96]. As described in Section 2.1.2, only data that include the matched intersection point in the foresight horizon (following 20 km) or history horizon (previous 5 km) are considered. The resulting window of new data points is marked as gray shaded area in Fig. A1.

C

Figure A2 shows the feature importance for the regression and quantile regression reference models for each horizon. The features include the current value for ambient temperature (T_a), battery temperature (T_b), and lower and upper cooling threshold ($T_{b,\,cool,\,lower/upper}$), as well as the minimum (min), maximum (max), mean value, and standard deviation (std) of the velocity (v) and the difference in velocity (Δv).

D

The point-forecast metrics are averaged over the four horizons and shown in Table A3. Two additional columns include the maximum absolute difference between predictions and true values for 90% and 95% of the test data. In case of the quantile prediction models, the metrics are calculated for the median prediction.

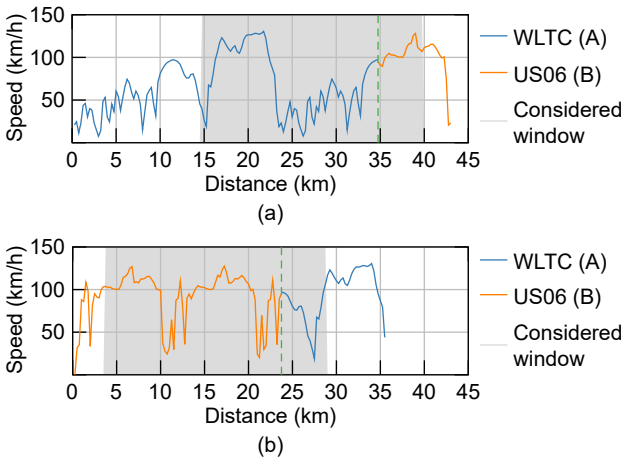

Fig. A1 Speed profiles of two augmented profiles using IBA with 3% tolerance for simulations of WLTC (A) and US06 (B) cycle. The two profiles A and B can be assembled by composition of the simulations around the matched intersection point (dashed line) as (a) AB and (b) BA.

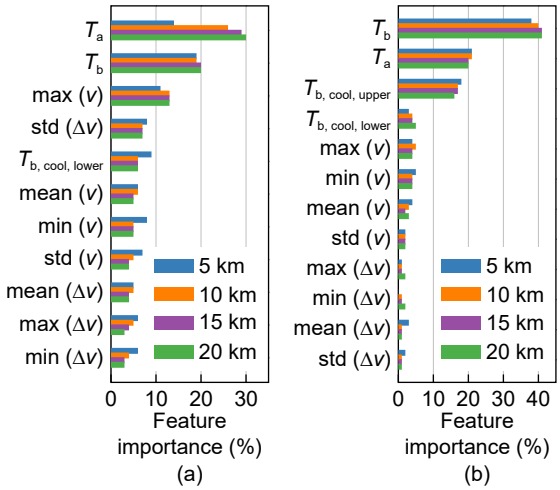

Fig. A2 Feature importance from (a) LGBM-ETR and (b) QETR reference models for all considered horizons.

CRediT

Andreas M. Billert: conceptualization, methodology,

Table A2 Hyperparameters and validation loss of the trials with lowest validation loss. Further descriptions of the hyperparameters are provided in Section 2.2.1.

Model	Parameter						
	Batch size	Base node number	Deeper network	Dropout rate	α_{MSE}	Learning rate	Total loss ($\times 10^{-4}$)
QCNN	32	256	False	0.2	0.01	3.50×10^{-5}	3.16
QGRU	80	128	False	0.2	0.01	1.01×10^{-4}	3.64
QLSTM	128	224	False	0.2	0.01	6.63×10^{-4}	3.23

Table A3 Point-forecast metrics on the test data, average over all four horizons and sorted from best to worst.

Model	RMSE (°C)	R^2	0.9 (°C)	0.95 (°C)
QGRU	0.66	0.84	1.00	1.31
QCNN	0.73	0.80	1.04	1.43
QLSTM	0.74	0.80	1.09	1.43
LGBM-ETR	1.09	0.59	1.52	2.33
QETR	1.26	0.46	1.69	2.60
QCNN22	1.30	0.47	2.05	2.47
SDT	1.37	0.39	2.00	2.65

software, investigation, writing original draft, and editing; Runyao Yu: methodology and software of time-series clustering; Stefan Erschen: reviewing; Michael Frey: reviewing; Frank Gauterin: supervision.

Acknowledgment

We acknowledge support by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

References

[1] W. Choi, E. Yoo, E. Seol, M. Kim, and H. H. Song, Greenhouse gas emissions of conventional and alternative vehicles: Predictions based on energy policy analysis in south korea, *Applied Energy*, vol. 265, p. 114754, 2020.

[2] Y. Zheng, X. He, H. Wang, M. Wang, S. Zhang, D. Ma, B. Wang, and Y. Wu, Well-to-wheels greenhouse gas and air pollutant emissions from battery electric vehicles in china, *Mitigation and Adaptation Strategies for Global Change*, vol. 25, no. 3, p. 355–370, 2020.

[3] A. Enthaler, T. Weustenfeld, F. Gauterin, and J. Koehler, Thermal management consumption and its effect on remaining range estimation of electric vehicles, in *Proc. 2014 International Conference on Connected Vehicles and Expo* (*ICCVE*), Vienna, Austria, 2014, pp. 170–177.

[4] Z. Feng, J. Zhao, C. Guo, S. Panchal, Y. Xu, J. Yuan, R. Fraser, and M. Fowler, Optimization of the cooling performance of symmetric battery thermal management systems at high discharge rates, *Energy & Fuels*, vol. 37, no. 11, pp. 7990–8004, 2023.

[5] V. Talele, U. Moralı, M. S. Patil, S. Panchal, and K. Mathew, Optimal battery preheating in critical subzero ambient condition using different preheating arrangement and advance pyro linear thermal insulation, *Thermal Science and Engineering Progress*, vol. 42, p. 101908, 2023.

[6] A. M. Billert, S. Erschen, M. Frey, and F. Gauterin, Predictive battery thermal management using quantile convolutional neural networks, *Transportation Engineering*, vol. 10, p. 100150, 2022.

[7] S. Park and C. Ahn, Stochastic model predictive controller for battery thermal management of electric vehicles, in *Proc. 2019 IEEE Vehicle Power and Propulsion Conference* (*VPPC*), Hanoi, Vietnam, 2019, pp. 1–5.

[8] Y. Xie, C. Wang, X. Hu, X. Lin, Y. Zhang, and W. Li, An MPC-based control strategy for electric vehicle battery cooling considering energy saving and battery lifespan, *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14657–14673, 2020.

[9] Z. Deng, X. Hu, X. Lin, Y. Che, Le Xu, and W. Guo, Data-driven state of charge estimation for lithium-ion battery packs based on gaussian process regression, *Energy*, vol. 205, p. 118000, 2020.

[10] S. Park and C. Ahn, Model predictive control with stochastically approximated cost-to-go for battery cooling system of electric vehicles, *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4312–4323, 2021.

[11] J. Lopez-Sanz, C. Ocampo-Martinez, J. Alvarez-Florez, M. Moreno-Eguilaz, R. Ruiz-Mansilla, J. Kalmus, M. Graeber, and G. Lux, Thermal management in plug-in hybrid electric vehicles: A real-time nonlinear model predictive control implementation, *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7751–7760, 2017.

[12] T. Fischer, T. Kraus, C. Kirches, and F. Gauterin, Nonlinear model predictive control of a thermal management system for electrified vehicles using FMI, in *Proceedings of the 12th International Modelica Conference*, doi:10.3384/ecp17132255.

[13] C. Zhu, F. Lu, H. Zhang, and C. C. Mi, Robust predictive battery thermal management strategy for connected and automated hybrid electric vehicles based on thermoelectric parameter uncertainty, *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 6, no. 4, pp. 1796–1805, 2018.

[14] S. Zhao, A. M. Reza, S. Jing, and C. C. Mi, A two-layer real-time optimization control strategy for integrated battery thermal management and hvac system in connected and automated hevs, *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 6567–6576, 2021.

[15] S. Park and C. Ahn, Computationally efficient stochastic model predictive controller for battery thermal management of electric vehicle, *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8407–8419, 2020.

[16] R. Baveja, J. Bhattacharya, S. Panchal, R. Fraser, and M. Fowler, Predicting temperature distribution of passively balanced battery module under realistic driving conditions through coupled equivalent circuit method and lumped heat dissipation method, *Journal of Energy Storage*, vol. 70, p. 107967, 2023.

[17] J. Park and Y. Kim, Supervised-learning-based optimal thermal management in an electric vehicle, *IEEE Access*, vol. 8, pp. 1290–1302, 2020.

[18] Y. Liu and J. Zhang, Self-adapting intelligent battery thermal management system via artificial neural network based model predictive control, doi:10.1115/DETC2019-98205.

[19] Y. Liu and J. Zhang, Electric vehicle battery thermal and cabin climate management based on model predictive control, *Journal of Mechanical Design*, vol. 143, no. 3, p. O31705 , 2020.

[20] P. Engel, S. Lempp, A. Rausch, and W. Tegethoff, Improving thermal management of electric vehicles by

prediction of thermal disturbance variables, in *Proc. ADAPTIVE 2018*, A. Rausch, C. Knieke, and M. Schranz, eds. Barcelona, Spain: IARIA, 2018, pp. 75–83.

[21] J. Park, Z. Chen, L. Kiliaris, M. L. Kuang, M. A. Masrur, A. M. Phillips, and Y. L. Murphey, Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion, *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 4741–4756, 2009.

[22] Y. N. Malek, M. Najib, M. Bakhouya, and M. Essaaidi, Multivariate deep learning approach for electric vehicle speed forecasting, *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 56–64, 2021.

[23] S. Fünfgeld, M. Holzapfel, M. Frey, and F. Gauterin, Stochastic forecasting of vehicle dynamics using sequential monte carlo simulation, *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 2, p. 1, 2017.

[24] A. M. Billert, M. Frey, and F. Gauterin, A method of developing quantile convolutional neural networks for electric vehicle battery temperature prediction trained on cross-domain data, *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 411–425, 2022.

[25] N. Moniz, P. Branco, and L. Torgo, Resampling strategies for imbalanced time series, in *Proc. 2016 IEEE International Conference on Data Science and Advanced Analytics* (*DSAA*), Montreal, Canada, 2016, pp. 282–291.

[26] A. L' Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, Machine learning with big data: Challenges and approaches, *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[27] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, Machine learning on big data: Opportunities and challenges, *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[28] M. Saripuddin, A. Suliman, S. Syarmila Sameon, and B. N. Jorgensen, Random undersampling on imbalance time series data for anomaly detection, in *Proc. MLMI 2021 The 4th International Conference on Machine Learning and Machine Intelligence*, Hangzhou, China, 2021, pp. 151–156.

[29] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham, Switzerland: Springer, 2017.

[30] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection: A survey, *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:57, 2009.

[31] A. Boukerche, L. Zheng, and O. Alfandi, Outlier detection: Methods, models, and classification, *ACM Comput. Surv.*, vol. 53, no. 3, pp. 55:1–55:37, 2020.

[32] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, A review on outlier/anomaly detection in time series data, *ACM Comput. Surv.*, vol. 54, no. 3, pp. 56:1–56:33, 2021.

[33] F. T. Liu, K. M. Ting, and Z.-H. Zhou, Isolation forest, in *Proc. 2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 413–422.

[34] R. J. Hyndman, E. Wang, and N. Laptev, Large-scale unusual time series detection, in *Proc. 2015 IEEE International Conference on Data Mining Workshop* (*ICDMW*), Atlantic City, NJ, USA, 2015, pp. 1616–1619.

[35] L. McInnes, J. Healy, N. Saul, and L. Großberger, Umap: Uniform manifold approximation and projection, *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[36] C. Weaver, A. C. Fortuin, A. Vladyka, and T. Albrecht, Unsupervised classification of voltammetric data beyond principal component analysis, *Chem. Commun.*, vol. 58, no. 73, pp. 10170–10173, 2022.

[37] S. Jia, H. Xianglin, Q. Sijun, and S. Qing, A bi-directional sampling based on k-means method for imbalance text classification, in *Proc. 2016 IEEE/ACIS 15th International Conference on Computer and Information Science* (*ICIS*), Okayama, Japan, 2016, pp. 1–5.

[38] G. Douzas, F. Bacao, and F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and smote, *Information Sciences*, vol. 465, pp. 1–20, 2018.

[39] S. Susan and A. Kumar, The balancing trick: Optimized sampling of imbalanced datasets—a brief survey of the recent state of the art, *Engineering Reports*, vol. 3, no. 4, p. e12298, 2021.

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[41] L. P. Silvestrin, L. Pantiskas, and M. Hoogendoorn, A framework for imbalanced time-series forecasting, in *Machine Learning, Optimization, and Data Science*, G. Nicosia, V. Ojha, E. La Malfa, G. La Malfa, G. Jansen, P. M. Pardalos, G. Giuffrida, and R. Umeton, eds. Springer International Publishing, 2022, pp. 250–264.

[42] C. Hou, J. Wu, B. Cao, and J. Fan, A deep-learning prediction model for imbalanced time series data forecasting, *Big Data Mining and Analytics*, vol. 4, no. 4, pp. 266–278, 2021.

[43] N. Moniz, P. Branco, and L. Torgo, Resampling strategies for imbalanced time series forecasting, *International Journal of Data Science and Analytics*, vol. 3, no. 3, pp. 161–181, 2017.

[44] X. Chen, L. Gupta, and S. Tragoudas, Improving the forecasting and classification of extreme events in imbalanced time series through block resampling in the joint predictor-forecast space, *IEEE Access*, vol. 10, pp. 121048–121079, 2022.

[45] C. Liu, J. Tan, H. Shi, and X. Wang, Lithium-ion cell screening with convolutional neural networks based on two-step time-series clustering and hybrid resampling for imbalanced data, *IEEE Access*, vol. 6, pp. 59001–59014, 2018.

[46] C. Li, L. Minati, K. K. Tokgoz, M. Fukawa, J. Bartels, A. Sihan, K.-I. Takeda, and H. Ito, Integrated data augmentation for accelerometer time series in behavior recognition: Roles of sampling, balancing, and fourier surrogates, *IEEE Sensors Journal*, vol. 22, no. 24, pp. 24230–24241, 2022.

[47] T. Schreiber and A. Schmitz, Surrogate time series, *Physica D*: *Nonlinear Phenomena*, vol. 142, no. 3, pp. 346–382, 2000.

[48] B. K. Iwana and S. Uchida, An empirical survey of data augmentation for time series classification with neural networks, *PloS one*, vol. 16, no. 7, pp. 1–32, 2021.

[49] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, Time series data augmentation for deep learning: A survey, in *Proc. of the Thirtieth International Joint Conference on Artificial Intelligence,* doi:10.48550/

arXiv.2002.12478.

[50] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, Generating synthetic time series to augment sparse datasets, in *Proc. 2017 IEEE International Conference on Data Mining* (*ICDM*), New Orleans, LA, USA, 2017, pp. 865–870.

[51] S. Demir, K. Mincev, K. Kok, and N. G. Paterakis, Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting, *Applied Energy*, vol. 304, p. 117695, 2021.

[52] B. Fu, F. Kirchbuchner, and A. Kuijper, Data augmentation for time series: Traditional vs generative models on capacitive proximity time series, in *Proc. of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, Corfu, Greece, 2020, pp. 107–116.

[53] B. Liu, Z. Zhang, and R. Cui, Efficient time series augmentation methods, in *Proc. 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics* (*CISP-BMEI*), Chengdu, China, 2020, pp. 1004–1009.

[54] B. K. Iwana and S. Uchida, Time series data augmentation for neural networks by time warping with a discriminative teacher, in *Proc. 2020 25th International Conference on Pattern Recognition* (*ICPR*), Milan, Italy, 2021, pp. 3558–3565.

[55] S. Kim, N. H. Kim, and J.-H. Choi, Prediction of remaining useful life by data augmentation technique based on dynamic time warping, *Mechanical Systems and Signal Processing*, vol. 136, p. 106486, 2020.

[56] M. Längkvist, L. Karlsson, and A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[57] J.-S. Ang, K.-W. Ng, and F.-F. Chua, Modeling time series data with deep learning: A review, analysis, evaluation and future trend, in *Proc. 2020 8th International Conference on Information Technology and Multimedia* (*ICIMU*), Selangor, Malaysia, 2020, pp. 32–37.

[58] B. Lim and S. Zohren, Time-series forecasting with deep learning: a survey, *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[59] Z. Liu, Z. Zhu, J. Gao, and C. Xu, Forecast methods for time series data: A survey, *IEEE Access*, vol. 9, pp. 91896–91912, 2021.

[60] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, A review of deep learning models for time series prediction, *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7833–7848, 2021.

[61] P.-P. Phyo, Y.-C. Byun, and N. Park, Short-term energy forecasting using machine-learning-based ensemble voting regression, *Symmetry*, vol. 14, no. 1, p. 160, 2022.

[62] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, An empirical comparison of machine learning models for time series forecasting, *Econometric Reviews*, vol. 29, nos. 5&6, pp. 594–621, 2010.

[63] C. M. Liapis, A. Karanikola, and S. Kotsiantis, Energy load forecasting: Investigating mid-term predictions with ensemble learners, in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez, eds. Cham, Switzerland: Springer International Publishing, 2022, pp. 343–355.

[64] C. M. Liapis and S. Kotsiantis, Energy balance forecasting: An extensive multivariate regression models comparison, in *Proc. of the 12th Hellenic Conference on Artificial Intelligence*, Corfu, Greece, 2022, pp. 1–7.

[65] M. Ali, Pycaret: An open source, low-code machine learning library in python, https://www.pycaret.org, 2023.

[66] S. B. Taieb and A. F. Atiya, A bias and variance analysis for multistep-ahead time series forecasting, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 62–76, 2016.

[67] J. Hong, H. Zhang, and X. Xu, Thermal fault prognosis of lithium-ion batteries in real-world electric vehicles using self-attention mechanism networks, *Applied Thermal Engineering*, vol. 226, p. 120304, 2023.

[68] J. Hong, Z. Wang, W. Chen, and Y. Yao, Synchronous multi-parameter prediction of battery systems on electric vehicles using long short-term memory networks, *Applied Energy*, vol. 254, p. 113648, 2019.

[69] Z. Deng, X. Lin, J. Cai, and X. Hu, Battery health estimation with degradation pattern recognition and transfer learning, *Journal of Power Sources*, vol. 525, p. 231027, 2022.

[70] P. Li, Z. Zhang, R. Grosu, Z. Deng, J. Hou, Y. Rong, and R. Wu, An end-to-end neural network framework for state-of-health estimation and remaining useful life prediction of electric vehicle lithium batteries, *Renewable and Sustainable Energy Reviews*, vol. 156, p. 111843, 2022.

[71] X. Song, F. Yang, D. Wang, and K.-L. Tsui, Combined CNN-LSTM network for state-of-charge estimation of lithium-ion batteries, *IEEE Access*, vol. 7, pp. 88894–88902, 2019.

[72] L. Ren, J. Dong, X. Wang, Z. Meng, L. Zhao, and M. J. Deen, A data-driven auto-cnn-lstm prediction model for lithium-ion battery remaining useful life, *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3478–3487, 2021.

[73] M. Kumar, Scikit-garden: A garden for scikit-learn compatible trees, https://github.com/scikit-garden/scikit-garden, 2013.

[74] D. Gan, Y. Wang, S. Yang, and C. Kang, Embedding based quantile regression neural network for probabilistic load forecasting, *Journal of Modern Power Systems and Clean Energy*, vol. 6, no. 2, pp. 244–254, 2018.

[75] W. Zhang, H. Quan, and D. Srinivasan, An improved quantile regression neural network for probabilistic load forecasting, *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 4425–4434, 2019.

[76] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, A multi-horizon quantile recurrent forecaster, in *Proc. 31st Annual Conference on Neural Information Processing Systems* (*NIPS*): *Time Series Workshop*, doi: 10.48550/arXiv.1711.11053.

[77] M. Lopez-Martin, A. Sanchez-Esguevillas, L. Hernandez-Callejo, J. I. Arribas, and B. Carro, Additive ensemble neural network with constrained weighted quantile loss for

probabilistic electric-load forecasting, *Sensors*, vol. 21, no. 9, p. 2979, 2021.
[78] C. Xu, C. Li, and X. Zhou, Interpretable lstm based on mixture attention mechanism for multi-step residential load forecasting, *Electronics*, vol. 11, no. 14, p. 2189, 2022.
[79] H. Zhang, W. Tang, W. Na, P.-Y. Lee, and J. Kim, Implementation of generative adversarial network-cls combined with bidirectional long short-term memory for lithium-ion battery state prediction, *Journal of Energy Storage*, vol. 31, p. 101489, 2020.
[80] M. Feurer and F. Hutter, Hyperparameter optimization, in *Automated Machine Learning*: *Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, eds. Cham, Switerland: Springer International Publishing, 2019, pp. 3–33.
[81] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, Hyperparameter optimization for machine learning models based on bayesian optimizationb, *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.
[82] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks, *IEEE Access*, vol. 8, pp. 52588–52608, 2020.
[83] T. T. Joy, S. Rana, S. Gupta, and S. Venkatesh, Hyperparameter tuning for big data using bayesian optimisation, in *Proc. 2016 23rd International Conference on Pattern Recognition* (*ICPR*), Cancun, Mexico, 2016, pp. 2574–2579.
[84] L. Wang, M. Feng, B. Zhou, B. Xiang, and S. Mahadevan, Efficient hyper-parameter optimization for nlp applications, in *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 2112–2117.
[85] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.
[86] S. Falkner, A. Klein, and F. Hutter, Bohb: Robust and efficient hyperparameter optimization at scale, in *Proc. of the 35th International Conference on Machine Learning*, doi: 10.48550/arXiv.1807.01774.
[87] J. Kong, W. Kowalczyk, D. A. Nguyen, T. Bäck, and S. Menzel, Hyperparameter optimisation for improving classification under class imbalance, in *Proc. Switerland 2019 IEEE Symposium Series on Computational Intelligence* (*SSCI*), Xiamen, China, 2019, pp. 3072–3078.
[88] D. A. Nguyen, A. V. Kononova, S. Menzel, B. Sendhoff, and T. Bäck, An efficient contesting procedure for automl optimization, *IEEE Access*, vol. 10, pp. 75754–75771, 2022.
[89] X.-B. Jin, W.-Z. Zheng, J.-L. Kong, X.-Y. Wang, Y.-T. Bai, T.-L. Su, and S. Lin, Deep-learning forecasting method for electric power load via attention-based encoder-decoder with bayesian optimization, *Energies*, vol. 14, no. 6, p. 1596, 2021.
[90] K. Miao, Q. Hua, and H. Shi, Short-term load forecasting based on cnn-bilstm with bayesian optimization and attention mechanism, in *Parallel and Distributed Computing, Applications and Technologies*, Y. Zhang, Y. Xu, and H. Tian, eds. Cham, Switerland: Springer International Publishing, 2021, pp. 116–128.
[91] A. Vysala and J. Gomes, Evaluating and validating cluster results, in Proc. *9th International Conference on Data Mining & Knowledge Management Process* (*CDKP 2020*), Toronto, Canada, 2020, pp. 37–47.
[92] H. Wang, J. Zhao, B. Wang, L. Tong, and G. Yuan, A quantum approximate optimization algorithm with metalearning for maxcut problem and its simulation via tensorflow quantum, *Mathematical Problems in Engineering*, vol. 2021, p. 6655455, 2021.
[93] V. Kuleshov, N. Fenner, and S. Ermon, Accurate uncertainties for deep learning using calibrated regression, in *Proc. of the 35th International Conference on Machine Learning*, doi: 10.48550/arXiv.1807.00263.
[94] M. Sajjad, Z. A. Khan, A. Ullah, T. Hussain, W. Ullah, M. Y. Lee, and S. W. Baik, A novel cnn-gru-based hybrid approach for short-term residential load forecasting, *IEEE Access*, vol. 8, pp. 143759–143768, 2020.
[95] M. Tutuianu, P. Bonnel, B. Ciuffo, T. Haniu, N. Ichikawa, A. Marotta, J. Pavlovic, and H. Steven, Development of the world-wide harmonized light duty test cycle and a possible pathway for its introduction in the european legislation, *Transportation Research Part D*: *Transport and Environment,* vol. 40, pp. 61–75, 2015.
[96] United States Environmental Protection Agency, Dynamometer drive schedules, https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules, 2022.

Andreas M. Billert received the BS degree in engineering science and the MS degree in automotive technology from TUM, Germany in 2017 and 2020, respectively, with visiting studies at University of California Berkeley, USA, and Shanghai Jiao Tong University, China. Currently, he is a PhD candidate cooperatively at Bayerische Motoren Werke (BMW) AG and Karlsruhe Institute of Technology (KIT) Institute of Vehicle System Technology, Germany, by supervision of F. Gauterin. His research interests include high-voltage batteries, predictive thermal management, machine learning, and neural networks.

Runyao Yu received the BS degree in electrical engineering and information technology from Technical University of Kaiserslautern, Germany in 2020, and the MS degree in electrical engineering and information technology from Technical University of Munich (TUM) , Germany in 2022, with his masters thesis at Bayerische Motoren Werke (BMW) AG under supervision of A. M. Billert. From 2020 to 2022, he was a research assistant at ETH Zurich and TUM. His research interests are predictive thermal and battery lifetime management, data science, and machine learning.

Stefan Erschen received the diploma degree (combined degree of BS and MS) in mechanical engineering from TUM, Germany in 2010. In 2018 he received the PhD degree in engineering from TUM, Germany. He works at BMW Group for more than 10 years in various fields, such as chassis design and software function development. His research interests are machine learning, optimization and robust design methodologies.

Frank Gauterin received the diploma degree (combined degree of BS and MS) in physics in 1989 from University of Münster, Germany, and the Dr. rer. nat. (PhD) degree from the University of Oldenburg, Germany in 1994. During 1989–2000 he worked as an acoustics engineer, and 2000–2006 as the director of the Noise Vibration & Harshness (NVH) Engineering Profit Centre at Continental AG, Hanover, Germany with locations in Germany, USA, and Malaysia. He is a full professor in vehicle technology at Karlsruhe Institute of Technology (KIT), Institule of Vehicle System Technology, Germany. He is the head of the Institute of Vehicle System Technology, and scientific spokesperson of the KIT Centre Mobility Systems. His research interests are vehicle conception, control, suspension and drive systems, tire road interaction, and NVH.

Michael Frey received the diploma degree (combined degree of BS and MS) and PhD degree in mechanical engineering from University of Karlsruhe, Geramny in 1993 and 2004, respectively. He is the manager of the research group automation and the research group suspension and propulsion systems at the Institute of Vehicle System Technology, Germany. His research interests are autonomous driving, driver assistance systems, operational strategies, suspension systems, vehicle dynamics, as well as vehicle modeling and optimization.