Development of an AI-based Energy Prediction for Electric Vehicles

Yannick Rauch[1], Nico Drobe[2], Tuyen Nguyen[1], and Reiner Kriesten[1]

[1] Karlsruhe University of Applied Sciences
Institute of Energy Efficient Mobility
yannick.rauch@h-ka.de, tuyen.nguyen@h-ka.de, reiner.kriesten@h-ka.de
[2] Karlsruhe University of Applied Sciences
Mechanical Engineering and Mechatronics

Abstract. The rise of electric vehicles offers new challenges as increasing energy efficiency and electric range through energy management systems as well as avoiding range anxiety by providing reliable range information. To realise this, predictive approaches based on a calculation of the expected energy consumption for the route to be driven are a feasible option. However, such approaches, realised through appropriate algorithms, usually require significant computing time, which can hinder their application. Therefore, this paper presents an approach to realise energy predictions using artificial intelligence (AI) approaches. The training of these AI models is performed with simulated data, generated by the algorithm to be replaced. Three AI models are build, trained, evaluated and optimised to predict a vehicle's energy consumption. A feed forward (FNN) and a recurrent neural network (RNN) model utilise deep learning approaches while a *XGBoost* model represents conventional machine learning techniques. In conclusion, the deep learning models struggle to match the results of the reference prediction algorithm, while the RNN model even fails to reduce calculation times. In contrast, the *XGBoost* model is able to generates accurate energy predictions, while drastically reducing the calculation time.

Keywords: Electric Vehicles, Energy Prediction, Energy Management

1 Introduction

With an increased awareness for energy and resource efficiency as well as stricter emission regulations a rise of hybrid- and all-electric vehicles occurred over the past years [1],[2],[3]. Energy management systems can provide smart solutions for the operation of these vehicles [4],[5]. Accordingly, increased electric ranges and energy efficiency can be achieved as well as a avoiding range anxiety by providing reliable range information [6]. To achieve this, predictive approaches that rely on prior knowledge of a route are a suitable solution [7],[8]. The resulting energy prediction can be used for the calculation of a route specific strategy to control energy consumption/distribution and/or to enable vehicle users to evaluate range capabilities to mitigate their range anxiety. For both use cases, a prediction must provide results with sufficient accuracy in acceptable computation time. Calculation methods based on the previous energy consumption from historical data can be used in short calculation times, but do not provide precise results. Empirical prediction methods can also be used, but require large and extensive data sets that are not widely available [8]. This leaves the use of physical and technical descriptions to create a model based prediction approach. However, these approaches to route

specific energy prediction cannot be computed in sufficient time and are therefore not applicable [9],[10]. This is especially the case when detailed models of a vehicle are used, which leads to an application of relatively simple calculation methods. In contrast, the best possible prediction of energy consumption for a specific route requires extensive and detailed models.

For a general applicability, the prediction should be able to provide detailed results on total energy consumption to derive range information and mitigate range anxiety. Furthermore the calculation of a predicted energy consumption profile, may be required, e.g. for implementing an energy control system. Accordingly, the energy prediction must calculate sufficiently accurate profiles of required data as well as overall correct totals, such as the total energy consumption. However, this must be done in acceptable calculation times so that the prediction can be integrated into a route planning process or used by energy management methods. In our context, a prediction algorithm based on an physical/technical model of vehicle, environment and driver is currently used to calculate the required energy consumption in the course of a route [8]. This algorithm provides sufficiently accurate results, but requires several seconds of computation time for execution and may therefore be too slow for an implementation in real vehicles or for acceptable user interaction.

Section 2 presents the approach pursued as well as the concept and developing method for an energy prediction artificial intelligence (AI). Subsequently, section 3 describes the implementation of AI energy prediction, focusing on evaluation and optimisation. In Section 4 the resulting models are examined based on three test scenarios. To conclude, Section 5 summarises this paper and provides a preview to future work.

2 Simulation Based Development of the Energy Prediction *AI*

Instead of the existing physical/technical based prediction algorithm, the introduced approach considers the use of machine learning methods to calculate the required prediction results, e.g. the energy consumption. This approach requires only an initial route specification to predict the corresponding energy consumption and is presented in Fig. 1.

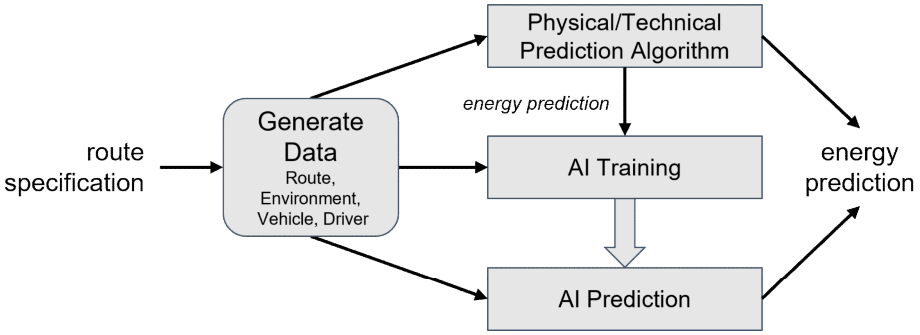

Fig. 1. Approach towards the use of machine learning for the prediction of a vehicle's energy consumption.

The input data for the *Artificial Intelligence* (AI) model is provided by a route information calculation [11], while the corresponding energy prediction for the AI training is provided by an energy and dynamic simulation algorithm [8]. Thus, the machine learning model is trained to replicate the simulated results and is expected to require significantly

less computing time than the calculation of the previously used prediction algorithm. Therefore, this approach uses a *MATLAB* implementation of the prediction algorithm. The required route data for this is generated by a self-developed *MATLAB* tool [11], that allows to provide various route information by accessing corresponding Web-*APIs* (*Application Programming Interfaces*). This information includes, but is not limited to, the geographical route date, gradients, traffic facilities and weather information. Based on this and a dataset for the vehicle and the driver, the physical/technical prediction algorithm generates the energy consumption profile over the course of a route. In order to reduce the dataset required to evaluate the general approach rather than producing a fully developed prediction, the driver and vehicle type is fixed and only the route and environment data are varied.

2.1 Development Method and Selection of the *AI* Models

To develop an *AI* model for energy prediction the method displayed in Fig. 2 is used. The necessary input data and energy prediction is generated by the algorithms implemented in *MATLAB* and then used for analysis and pre-processing according to the requirements of the chosen *AI* model. Subsequently the *AI* training loop is carried out, by running, evaluating and optimising the *AI* model.

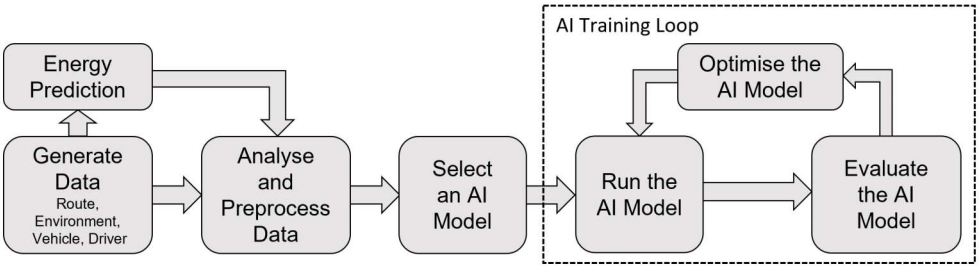

Fig. 2. Method for a simulation based development of a machine learning approach for energy prediction.

Data Generation and Preprocessing Through preprocessing and analysis of the generated data, representative datasets are created to be used in model training. The data generation process includes reading the generated input data and corresponding prediction results to normalise, standardise and categorise them to ensure their suitability for the *AI* models. The subsequent analysis of the data includes a plausibility check that assesses overall quality and integrity to ensure reliability of subsequent analyses and model training. In addition, insight into the characteristics of the data is gained for further development, e.g. through histograms, to understand distribution patterns.

Model Selection As a basic model, a feed forward neural network (*FNN*) is selected based on its successful use in various prediction tasks across different domains and its ability to handle complex relationships [12]. As the prediction of energy consumption involves intricate interactions between multiple variables, the capacity of a *FNN* to learn and model such complex patterns becomes particularly advantageous. With the availability of deep learning frameworks, e.g. *PyTorch* or *TensorFlow*, the implementation is also not that complex. In addition, the preparation of the training data is much easier compared to a recurrent neural network (*RNN*), which will also be used. This is due to

its specific design to capture dependencies within sequential data, which is the case for energy and dynamic behaviour of a vehicle. In the context of predicting energy consumption, which mainly results from the dynamics and is influenced by past states, the ability of a *RNN* to retain and use information from previous inputs seems promising.

While the models already presented use deep learning methods, *XGBoost* was selected as a third model, which is based on decision trees and ensemble learning. This model offers efficiency and scalability making it suitable for dealing with large datasets and computationally intensive tasks. *XGBoost* can efficiently handle the complexity and deliver fast and reliable predictions, which is proven successful in various machine learning competitions and real-world applications. Its robustness and track record make it a popular choice for tackling diverse prediction problems, which was also demonstrated across a variety of domains [13].

2.2 Development of the *AI* Model

After selecting the *AI* models to be studied for their ability to predict the energy consumption of an electric vehicle, the training of these models is to be carried out. For this purpose, a framework for development has to be selected first. The training is realised by *AI Training Loop*, displayed in Fig. 2, which is executed through two steps. The first execution is done during the process of model building and training. The second loop run is performed with the results of an already trained *AI* model, which are to be evaluated and may lead to an optimisation of the model.

Framework To build an *AI* model the selection of an adequate framework ensures the efficiency of the model development. For deep learning approaches *PyTorch* and *TensorFlow* are among the most widely used frameworks and have gained significant popularity and a large user base [14]. For this work the implementation of the *FNN* and *RNN* models uses *PyTorch* as framework, due to its similarities to *Python* which is already used for preprocessing, while the *XGBoost* library itself provides all necessary resources for building the model.

Building and Training For training, evaluation and optimisation of a model the dataset is split into a training and validation set. This allows the calculation of an appropriate loss to prevent a bias towards the dataset used for training [15]. The choice of training to validation split ratio depends on the specific requirements and objectives of the project. In this case, a ratio of 70:30 is used to create a balance between a sufficient amount of data for training and an extensive amount of evaluation. This balance is crucial to avoid overfitting, where the model becomes too specialised to the training data and fails to generalise well to new instances not yet seen. Additionally, the data is randomly shuffled to achieve more equitable representation of different driving scenarios within the training and test datasets, which ensures that the model learns from a diverse range of situations. Furthermore, to accelerate the training process, scaling of the features is applied before finally training the *AI* model with the given dataset. For the implementation of *FNN* and *RNN* models, *ReLU* is used as activation function and *Adam* is used as optimiser.

Evaluation For evaluation of the trained *AI* model two methods are used. As a first step, the hold-out method is used to check whether the model generalises well. This involves

a single instance of data split to training and validation datasets as already described. As a second step cross-validation is used to gain a more comprehensive understanding of the model's performance. This involves multiple datasets of differently split training and validation data, which are used in the training process. To further evaluate the generalisation capabilities of the trained models several test scenarios are created that the model has not yet seen. For this, scenarios are created over different environments, including city, countryside, and highway settings. To quantify the discrepancy between the *AI* predicted and reference energy consumption values, the mean square error (*MSE*) and the total energy consumption of a route are used. Furthermore, a comparison of the computation time between the chosen *AI* models and the *MATLAB* simulation tool is performed to assess the calculation speed and efficiency.

Optimisation To optimise the *AI* models, both a systematic approach in terms of hyperparameters and a manual approach of feature engineering are considered. The hyperparameter optimisation is performed with a grid search algorithm, which iterates a set of parameters defined by the developer and evaluates the model performance for each variant. For optimisation the *GridSearchCV* algorithm is used, which combines cross-validation with hyperparameter tuning [16]. Accordingly, this brute force search can find the hyperparameter set that performs best over all possible parameter combinations, also taking into account the generalisation of the model. Furthermore, a optimisation of the model is to be achieved by adding features created by the developer, which are not presented in the given dataset but seem promising to increase the model performance.

3 Implementation

To develop the chosen *AI* models for energy prediction an implementation for training and executing these models is required. Furthermore, corresponding representative datasets are required. Therefore, test scenarios are defined and the resulting data is analysed to ensure data quality. Finally the features and labels for the creation of the *AI* models are defined so that evaluation and optimisation can be carried out.

3.1 Data Generation and Analysis

As a first step for building and training of the *AI* models test scenarios are defined and corresponding data is generated by the given *MATLAB* tool. Based on this data it is crucial to examine the data for sanity and reasonability.

Data Generation For generating the required test scenarios a variety of routes is created across various driving scenarios, including city, countryside, and highway routes, to provide diverse and representative conditions. The dataset should contain a diverse specification of route properties, such as road gradient, number of traffic lights, weather and wind conditions, season and daytime conditions as well as tyre configurations.

Sanity Checks and Histogram Analysis The examination of the generated data revealed some problems with the functionality of the *MATLAB* tool used, particularly related to its use for generating representative data sets for training *AI* models. For example the tool generates weather related data according to the actual conditions. However, a histogram analysis of these generated datasets shows that this does not lead

to a proper distribution of weather related data. Accordingly, the tool was adjusted to manually set weather conditions to achieve a diverse distribution of weather data. Another finding was that generated times for sunrise, sunset and actual time are difficult to interpret and therefore not suited well as features. However, this data is only used by the prediction algorithm to determine the daytime. As a consequence, the tool was changed to manually setting the daytime to be used as a feature. In addition, it is also important to check the distribution of the data, as was done for the given curve radius data. A noteworthy observation in this data plot is a significant concentration of data at a value of 5000 m, which indicates that the steering wheel is in a straight position. With the knowledge of this skewed distribution, an optimisation approach can be carried out.

Selection of Features and Labels After careful investigation on the provided data the features for building the *AI* model can be defined. The chosen features are tire type, ambient temperature, weather state, daytime, curve radius, gradient, air density, resulting wind speed, speed limits, tunnels and traffic signals. To represent the energy prediction capability, the electrical energy consumption is chosen as label.

3.2 Evaluation and Optimisation

After the successful building of the *AI* models, the evaluation and optimisation of these models are the next essential steps to improve overall performance and accuracy. The deep learning models will be evaluated using the loss curves calculated by the *MSE* and serve as a visual tool for understanding a model's learning behaviour.

Increasing Data Points For the first experiment, a *FNN* model, consisting of four hidden layers with 1024 neurons in each layer, a learning rate of 0.001 and 300 training epochs is implemented. This model is trained with a dataset of 100,000 data points and results in the loss curve displayed in Fig. 3.

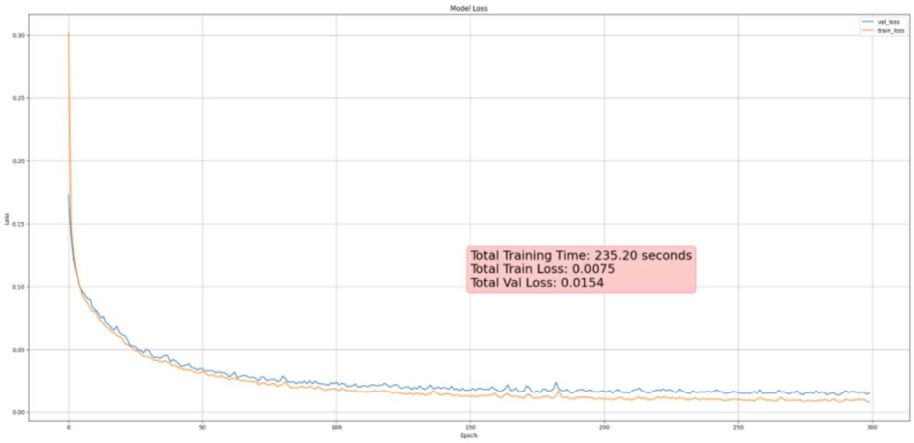

Fig. 3. Loss curve of a *FNN* experiment.

At first glance, the loss trend of the train and the validation data appears promising. However, for final evaluation the model was used to make a prediction on a previously unforeseen test scenario. This evaluation shows that the model falls short when applied to

previously unseen data, which could lead to the assumption that the model is overfitting. The corresponding loss curve in Fig. 3, however, has no clues, as the validation curve hardly deviates from the training curve. Therefore, the assumption is made that the number of training points is not sufficient.

Evaluation of the Amount of Training Data Establishing the precise number of data points necessary to build an accurate *AI* model presents a challenge since it highly depends on the specific problem at hand. Therefore, the *scikit*-learn *Python* library provides a method where different amounts of data points are tested via cross validation to investigate, whether an increase in data points leads to an improvement in the model's predictive accuracy. Fig. 4 presents a visual representation of this method, applied to an *XGBoost* model by using the *MSE* as a metric to evaluate the model's accuracy, using the total energy consumption as a target variable. The used *XGBoost* model consists of 100 decision trees with a maximum depth of 5 for each tree, while a learning rate of 0.1 and a 5-fold cross validation are used. This example shows, that between 50,000 and 150,000 data points the curve still drops significantly, while past the 300,000 data points mark, no significant changes are observable.

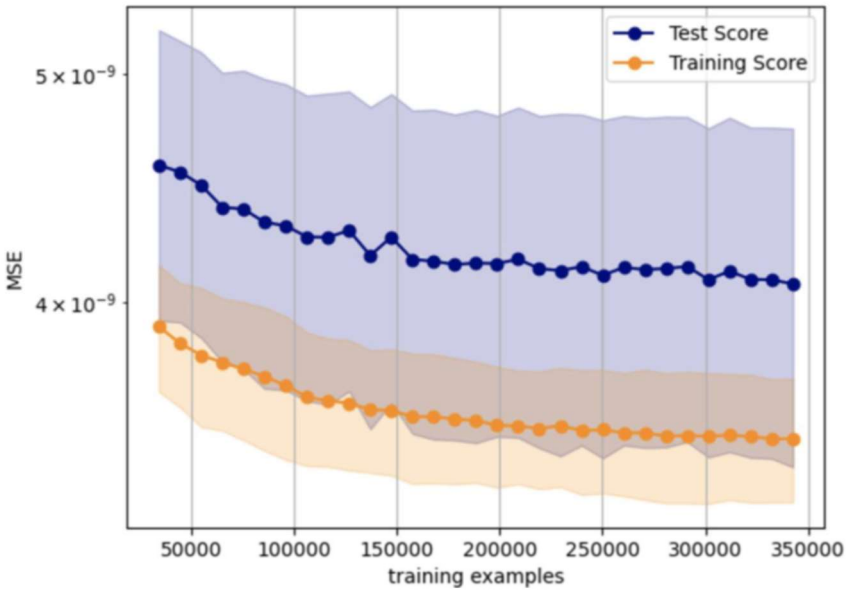

Fig. 4. *XGBoost* learning curve by evaluating different sizes of training examples.

Optimising with *GridSearchCV* To optimise the hyperparameter the *GridSearchCV* algorithm provided by *scikit*-learn is used. As this functions utilises brute-force iteration of the hyperparameters and combines this with an application of cross-validation it should be noted that it is a time intensive calculation. However, the optimisations that can be achieved with this are limited, especially compared to other optimisation approaches such as feature engineering. For example, when applied to an *FNN* model, the training loss is only improved by 10.68 % and the validation loss is only improved by 4.62 %, while the training time triples.

Feature Engineering The evaluation of the trained *AI* models as well as the knowledge of the used prediction algorithm shows that the representation of traffic signals can

have a significant influence on the resulting prediction. Therefore, a new feature called *distance_trafficSignals* is introduced, which describes the distance to the next traffic signal in order to represent the approaching behaviour of a vehicle. To evaluate, whether the new feature provides an advantage a *FNN* model consisting of six hidden layers with 1024 neurons in each layer, a learning rate of 0.0001 and 300 training epochs is used. With the inclusion of the new feature, a remarkable improvement of 36.72 % in training loss and an enhancement of 37.53 % in validation loss is achieved, while there was only a marginal increase in the total training time.

Optimising by Data Scaling Adjustment Based on the analysis of the histograms, it was suggested that different methods may be useful for scaling the data. Accordingly, two experiments are conducted to determine whether the use of a quantile transformation leads to an improvement in model accuracy compared to the use of standardisation. To carry this out a *FNN* model consisting of six two layers with 512 neurons in each layer, a learning rate of 0.0001 and 200 training epochs is used. This experiment shows, that the use of a quantile transformation as data scaling technique results in a reduction of 16.57 % of training loss and 11.97 % of validation loss.

Optimising the *RNN* model The evaluation and optimisation of the *RNN* model is of particular interest, as different sequence lengths have a significant impact on the performance. Accordingly, the sequence length is varied during the evaluation process to gain comprehensive insights how this factor influences the model's ability to learn, generalise, and predict. To speed up the experimentation, a simplified RNN model consisting of two layers with 128 neurons in each layer, a learning rate of 0.001 and 100 training epochs is used. As a baseline the *RNN* is essentially transformed to a *FNN* by using a sequence length of one and is compared to a RNN with a sequence length of six. This shows significant improvements regarding training and validation loss, while tripling the time required for training. A further increase to a sequence length of ten does not lead to another improvement but reveals issues with the simplified model. As a consequence the complexity of the *RNN* is increased to four hidden layers with 512 neurons in each layer, a learning rate of 0.0001 and 200 training epochs. Using this model, it can then be shown that a further increase in the sequence length does not offer improvements, which is why a sequence length of ten will be used for the *RNN* model.

3.3 Resulting Models

After applying the evaluation and optimisation activities to the models, this results in three descriptions for further use in discussion. To ensure a fair comparison between the deep learning models, the *FNN* and *RNN* model are configured with identical hyperparameters. The models consist of four hidden layers with 512 neurons in each layer, while training is done in 200 epochs with a learning rate of 0.0001. As already mentioned, the *RNN* uses a sequence length of 10. In contrast, the optimised *XGBoost* model employs different parameters It consists of 400 decision trees with a maximum depth of 18 for each tree and uses a learning rate of 0.1.

4 Evaluation and Discussion

The evaluation and discussion of the developed *AI* models is based on the application to 3 test scenarios that are unknown to the models. For evaluation the calculation time of

the prediction models as well as the accuracy of the resulting energy consumption profile and corresponding total are considered and compared with the originally used *MATLAB* prediction algorithm.

4.1 Test Scenarios

For evaluation of the energy prediction capabilities of the *AI* models three test routes representing a specific route type are used. The first scenario is an urban route with a length of approximately 5 km, 11 traffic lights and mainly driven at a speed of 50 km/h. It is simulated during the day with summer tyres under overcast skies and an ambient temperature of 11 °C. The second scenario is a route with a significant highway part, predominantly high speed sections over 100 km/h over a distance of around 26 km with only 6 traffic lights. The simulation is based on broken clouds during the day, an ambient temperature of 14 °C and winter tyres. A third scenario is used, representing a countryside journey, usually at a speed of over 70 km/h over a distance of approximately 22 km and significant differences in altitude between 633 and 1233 metres above sea level. The journey is simulated using winter tyres at an ambient temperature of 13 °C and scattered clouds during the day.

4.2 Evaluation

The energy prediction results of the *MATLAB* prediction algorithm used as a reference and the results of the three *AI* models are presented for each test scenario in Tables 1, 2 and 3. Across all three routes, the models produce different results. On the city route, the *FNN* slightly under predicts the energy consumption by approximately 7 % compared to *MATLAB*, while both *RNN* and *XGBoost* overpredict, with differences of 15.4 % (*RNN*) and 11.7 % (*XGBoost*) respectively. For the countryside route, the differences between *MATLAB* and the *AI* models are more significant. The *FNN* notably predicts negative energy consumption, while the *RNN* significantly underestimates it by 40.5 %. In contrast, *XGBoost* slightly overpredicts the energy usage by 5.2 %. On the highway route, all three models underestimate energy consumption to varying degrees, with the prediction of *XGBoost* again coming closest to the reference, but still underestimating by 13 %.

Table 1. Evaluation of characteristics of test scenario 1 (urban route).

	MATLAB Algorithm	***FNN***	***RNN***	***XGBoost***
MSE of Energy Consumption	-	0.069 Wh	0.064 Wh	0.059 Wh
Total Energy Consumption	0.466 kWh	0.434 kWh	0.538 kWh	0.521 kWh
Calculation Time	3.66 s	0.34 s	5.78 s	0.01 s

In general, *XGBoost* seems to offer more consistent results and appears to perform particularly well across different routes compared to the deep learning models. Furthermore, it is notable that for all three routes, the *XGBoost* model achieved the lowest *MSE* values, indicating that its predictions are closer to the reference values. In terms of calculation time, the *XGBoost* model again stands out among the three *AI* models. Not only does the model provide the most accurate predictions, it also has by far the shortest

calculation times. For all three routes, the calculation time to run the *XGBoost* model is measured in the order of hundredths of a second. In comparison, the *MATLAB* algorithm and the *RNN* model take several seconds and the calculation of the *FNN* model takes at least a few milliseconds.

Table 2. Evaluation of characteristics of test scenario 2 (highway route).

	MATLAB Algorithm	***FNN***	***RNN***	***XGBoost***
MSE of Energy Consumption	-	0.110 Wh	0.082 Wh	0.054 Wh
Total Energy Consumption	4.270 kWh	3.048 kWh	3.173 kWh	3.713 kWh
Calculation Time	7.75 s	1.36 s	29.57 s	0.03 s

Table 3. Evaluation of characteristics of test scenario 3 (countryside route).

	MATLAB Algorithm	***FNN***	***RNN***	***XGBoost***
MSE of Energy Consumption	-	0.160 Wh	0.077 Wh	0.038 Wh
Total Energy Consumption	2.341 kWh	-0.211 kWh	1.393 kWh	2.462 kWh
Calculation Time	6.93 s	1.17 s	22.81 s	0.02 s

When comparing the prediction of energy consumption profiles, Fig. 5 shows that all models have problems with the accurate prediction of energy peak values. In particular, the models tend to underestimate the peaks, especially in stop-and-go scenarios, such as red traffic lights. In some cases, the models also shift the energy peaks, indicating an incorrect prediction of the traffic light condition.

4.3 Discussion

In conclusion, the *XGBoost* model, which represents the conventional machine learning approach, achieves the best results in terms of predictive accuracy and calculation time. Regarding computational efficiency, the *FNN* model outperforms the *RNN* model, illustrating a trade-off between predictive accuracy and computational speed. However, it should also be noted that due to the recursive nature of the *RNN* model, calculation times increase drastically for longer distances. This issue disqualifies the use of the *RNN* model as a replacement of the *MATLAB* algorithm. Nevertheless, with the *FNN* model, there is a risk that the prediction is significantly inaccurate, such as in the prediction for the countryside scenario, which could also hinder the use of this approach.

5 Conclusion

With the motivation to use detailed models for predicting energy consumption for specified routes, an approach is pursued that uses machine learning to reproduce these results in less computing time. Towards this, two deep-learning models and one model based on

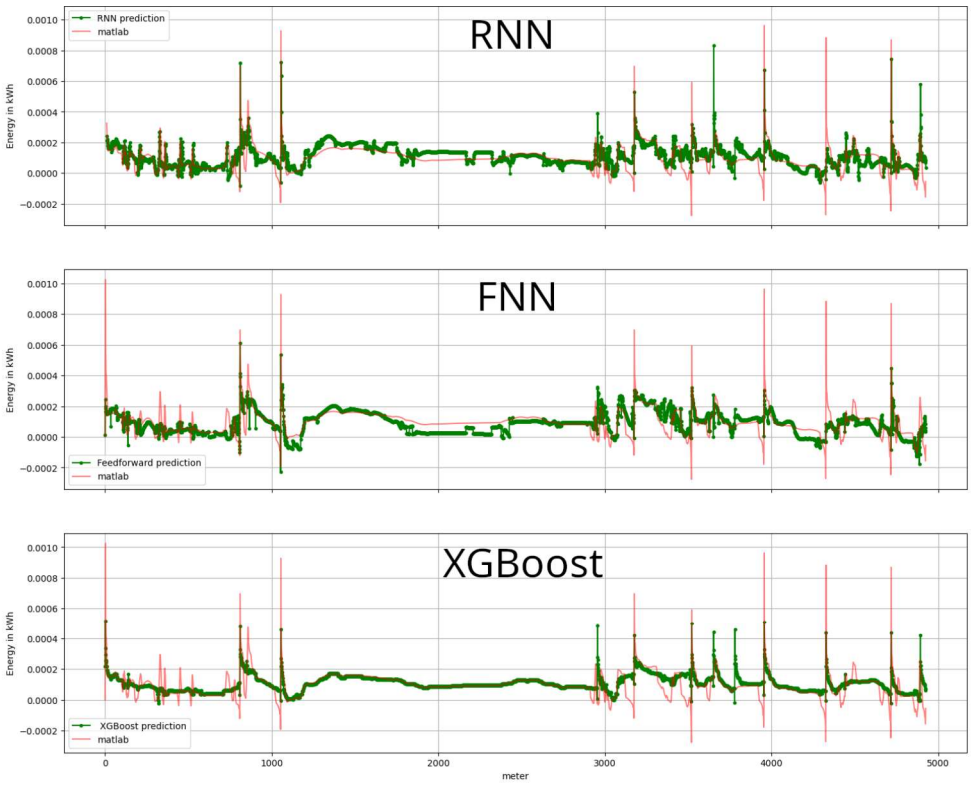

Fig. 5. Predicted energy consumption of test scenario 1 (city route) of the *MATLAB* prediction compared to the prediction of the *AI* models.

traditional machine learning are used. As deep learning models, both a *FNN* as a simple approach and a *RNN* model to account for the recursive nature of energy prediction are used. In addition, a *XGBoost* model represents a conventional machine learning approach using decision trees and ensemble learning methods. All three models are trained, evaluated and optimised to predict the electrical energy consumption of an electric vehicle based on given route and environment characteristics. The corresponding training data is generated by a *MATLAB* prediction algorithm, which is to be replaced by the *AI* models. However, the results show that the deep learning models struggle to match the results of the reference prediction algorithm. In contrast, the *XGBoost* model using a conventional approach successfully generates accurate energy predictions, while drastically reducing the calculation time. To conclude, it is shown that it is possible to make an energy prediction with an alternative method to the use of a physics-based algorithm.

The first assessment of the models is to be expanded in future work. On the one hand further evaluation is to be carried out, comparing the predictions of the *AI* models to measured energy consumption. On the other hand the scope of the *AI* models prediction can be expanded, e.g. by predicting further energy and dynamic quantities.

Acknowledgment

This work was carried out as part of the *move.more* project, sponsored by the Federal Ministry of Education and Research of the Federal Republic of Germany. The development was carried out by Nico Drobe in the context of his bachelor thesis supported by the company *innovex GmbH*, represented by Dominik Traxl and Silas Raschke.

References

[1] Kraftfahrtbundesamt. *Anzahl der Hybridautos in Deutschland von 2006 bis 2023.* Edited by Statista. 2023. URL: https://de.statista.com/statistik/daten/studie/265993/umfrage/anzahl-der-hybridautos-in-deutschland/ (visited on 07/14/2023).

[2] Kraftfahrtbundesamt. *Anzahl der Elektroautos in Deutschland von 2006 bis April 2023.* Edited by Statista. 2023. URL: https://de.statista.com/statistik/daten/studie/265995/umfrage/anzahl-der-elektroautos-in-deutschland/ (visited on 07/14/2023).

[3] BMU. *Das System der CO2-Flottengrenzwerte für Pkw und leichte Nutzfahrzeuge.* Edited by Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit. 2020.

[4] A. Basler. "Eine modulare Funktionsarchitektur zur Umsetzung einer gesamtheitlichen Betriebsstrategie für Elektrofahrzeuge". Institut für Fahrzeugsystemtechnik. Dissertation. Karlsruhe: Karlsruher Institut für Technologie, 2015. 246 pages.

[5] K. Çağatay Bayindir, M. A. Gözüküçük, and A. Teke. "A comprehensive overview of hybrid electric vehicle: Powertrain configurations, powertrain control techniques and electronic control units". In: *Energy Conversion and Management* 52.2 (2011), pages 1305–1313.

[6] K. Kruppok, C. Gutenkunst, R. Kriesten, and E. Sax. "Prediction of energy consumption for an automatic ancillary unit regulation". In: *17. Internationales Stuttgarter Symposium.* Edited by M. Bargende, H.-C. Reuss, and J. Wiedemann. Proceedings. Wiesbaden: Springer Fachmedien Wiesbaden, 2017, pages 41–56.

[7] K. Kruppok. *Analyse der Energieeinsparpotenziale zur bedarfsgerechten Reichweitenerhöhung von Elektrofahrzeugen.* ger. 1st ed. Kruppok, Kurt (VerfasserIn). Tübingen: expert verlag, 2020. 293 pages.

[8] C. Gutenkunst. "Prädiktive Routenenergieberechnung eines Elektrofahrzeugs". de. Dissertation. Karlsruhe: Karlsruher Institut für Technologie, 2020. 214 pages.

[9] R. Oberfell. "Stochastische Simulation von Energieflüssen im Nutzfahrzeug Ein einsatzorientiertes Bewertungs- und Optimierungsverfahren. Ein einsatzorientiertes Bewertungs- und Optimierungsverfahren". Institut für Fahrzeugsystemtechnik. Dissertation. Karlsruhe: Karlsruher Institut für Technologie, 2015. 344 pages.

[10] S. F. Tie and C. W. Tan. "A review of energy sources and energy management system in electric vehicles". In: *Renewable and Sustainable Energy Reviews* 20 (2013). PII: S1364032112006910, pages 82–102.

[11] T. Nguyen and Y. Rauch. "Real Route Generation for Simulation Based Developement". In: *Reports on Energy Efficient Mobility – Volume 2.* Edited by D. Feßler et al. 2nd edition. Karlsruhe: Zenodo, 2022, pages 58–64.

[12] S. Murat H. "A brief review of feed-forward neural networks". In: *Communications Faculty Of Science University of Ankara* 50.1 (2006). PII: commua1-2, pages 11–17.

[13] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System". In: 11.34 (2016). KDD'16 changed all figures to type1, pages 785–794.

[14] R. O'Connor. *PyTorch vs TensorFlow in 2023.* Edited by AssemblyAI Inc. 2021. URL: https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/ (visited on 10/13/2023).

[15] V. R. Joseph. "Optimal Ratio for Data Splitting". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 15.4 (2022). Statistical Analysis and Data Mining: The ASA Data Science Journal, 2022, pages 531–538.

[16] scikit-learn developers. *scikit GridSearchCV.* 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (visited on 10/14/2023).