PREDICTION OF TIME AND DISTANCE OF TRIPS USING EXPLAINABLE ATTENTION-BASED LSTMS

Ebrahim Balouji
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden

Jonas Sjöblom
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden

Nikolce Murgovski
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden

Morteza Haghir Chehreghani
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden

March 28, 2023

ABSTRACT

In this paper, we propose machine learning solutions to predict the time of future trips and the possible distance the vehicle will travel. For this prediction task, we develop and investigate four methods. In the first method, we use long short-term memory (LSTM)-based structures specifically designed to handle multi-dimensional historical data of trip time and distances simultaneously. Using it, we predict the future trip time and forecast the distance a vehicle will travel by concatenating the outputs of LSTM networks through fully connected layers. The second method uses attention-based LSTM networks (At-LSTM) to perform the same tasks. The third method utilizes two LSTM networks in parallel, one for forecasting the time of the trip and the other for predicting the distance. The output of each LSTM is then concatenated through fully connected layers. Finally, the last model is based on two parallel At-LSTMs, where similarly, each At-LSTM predicts time and distance separately through fully connected layers. Among the proposed methods, the most advanced one, i.e., parallel At-LSTM, predicts the next trip's distance and time with 3.99 % error margin where it is 23.89 % better than LSTM, the first method. We also propose TimeSHAP as an explainability method for understanding how the networks perform learning and model the sequence of information.

***K*eywords** Trip prediction · Attention LSTM · Parallel LSTM · Explainability · TimeSHAP

1 Introduction

Trip prediction constitutes an important element for intelligent transport in various ways, e.g., simulation of public transportation, traffic analysis, battery charge planning, demand and load investigation, and preparation of vehicles for trips [6–8, 10, 24]. It also helps to optimize energy management, save energy, prolong components' lifetime, and can also be used to support electric power systems [2, 3]. In particular, it yields increasing efficiency and reduces the energy consumption of Electric Vehicles (EVs) and even combustion engines.

Predicting the trip time can help prepare the vehicle for the mission by thermally conditioning the passengers' cabin. This is specifically important for batteries or engines to avoid a cold start. In an electric vehicle, cold start may cause a reduction in the efficiency leading to capacity degradation of batteries and, subsequently, a reduction in their lifetime [16–18, 22].

Distance prediction can help to estimate how much energy will be needed. Also, it can pinpoint if warming batteries before the trip is needed by trading off energy efficiency, battery lifetime, and power availability. Knowing the amount of power needed for the trip can help to optimize charging policies and avoid fast charging, which may be used to

take preventive measures for fast battery degradation [1, 23]. Using the information on both time and distance of trips can open the possibility of using batteries for the power flow in electric grid management [9, 11] and reduce energy consumption. It can also be used to plan to build charging stations in the cities, especially in megacities. Observing and predicting the general traveling pattern of vehicle users in a region can help to estimate the overall statistics of a transportation system.

Reaching all the accomplishments mentioned above highly relies on the possibility of predicting the accurate and exact time and distance of future travels. However, passengers do not always travel according to a linear planning schedule, and the trips can be made according to stochastic patterns. This stochasticity may even increase when predicting a fleet of vehicles. Several works have been proposed to predict the possible destination for a trip/vehicle [7, 10, 12, 13, 19] and the path/route to be taken under different conditions [2–4]. However, the prediction of the exact time of the trip and the distance that the vehicle will travel are not studied. In this paper, we develop two approaches to address this task. We propose predicting future trips' time and distance by analyzing historical data using unique features-specific learning models based on LSTM and attention-based LSTM (At-LSTM) neural networks. We also develop timeSHAP, a posthoc model-agnostic explainability method specifically tailored for LSTM and At-LSTM handling multidimensional and length-variant sequences of information.

The contributions of this paper can be summarized as follows: i) We develop a novel deep learning-based framework utilized by At-LSTM-based structure to predict future trip time and distance that car will travel. ii) Using the most advanced method, the prediction error significantly decreases by about $\approx 22\%$ compared to using only LSTM. iii) The proposed method has minimal preprocessing steps and is a data-driven approach with a diverse dataset obtained from 700 vehicles. iv) we also develop timeSHAP as a model-agnostic LSTM explainer that is utilized based on KernelSHAP and expands it to analyzing sequential data. The rest of the paper is organized as follows: In section 2, we introduce the data, the four different models as well as the explainability component. In section 3, we demonstrate the experiments and discuss the respective results. Finally, we conclude the paper in section 4.

2 Methodology

In this section, we describe four different methods, including two LSTM based and two At-LSTM-based networks, as four solutions to predict the time of future trips together with the possible distance that will be traveled by the vehicle.

2.1 Data and preprocessing

The dataset used in this study is a real-world dataset where 95962 trips have been collected by 700 GPS-tracked vehicles in Sweden. To avoid GPS disconnection, trips made within 10 min are considered one trip. Furthermore, the trips shorter than 3 km are discarded since the cold start of the vehicles are more energy efficient for short-distance travels. After applying such preprocessing steps, about 50% of the original trips and 59% of the vehicles are included in the filtered dataset. The density map of the trips based on GPS coordinates and the destination is illustrated in Fig. 1. More detail about the trips can be found in [15].

We define Δt as in how many seconds the next trip will be made, i.e.,

$$\Delta t_n = t_n - t_{n-1}, \tag{1}$$

where t_n and t_{n-1} are the time of the last two trips in historical data. We also define d_n as the distance the vehicle has traveled. Finally, to consider the possibility of variation of the trip frequency on different days of the week, we consider the corresponding weekday as a feature alongside Δt and d. Therefore prediction of the future trip time and distance by observing the historical data can be formulated as follows.

$$\begin{bmatrix} \Delta t_{n+1} \\ d_{n+1} \end{bmatrix} = f \begin{bmatrix} (\Delta t_{n-1}, \Delta t_{n-2}, \Delta t_{n-3}, \ldots, \Delta t_n) \\ (d_{n-1}, d_{n-2}, d_{n-3}, \ldots, d_n) \\ \text{(corresponding weekday of each trip)} \end{bmatrix}, \tag{2}$$

where f represents, for example, a deep learning model.

2.2 LSTM based methods

The first method (called $PM1$) consists of three LSTM layers followed by two fully connected layers (see Fig. 2 (a)). The input to the LSTM networks is a multidimensional matrix introduced in Eq. (2). The second method is based on using two parallel LSTM-based neural networks where each branch of the network receives separate Δt and d, and the outputs of the two LSTMs are then concatenated through two fully connected layers. Fig. 2 (b) illustrates the structure of the second method (called $PM2$). The third method (called $PM3$) is the *attention* version of the LSTM-based

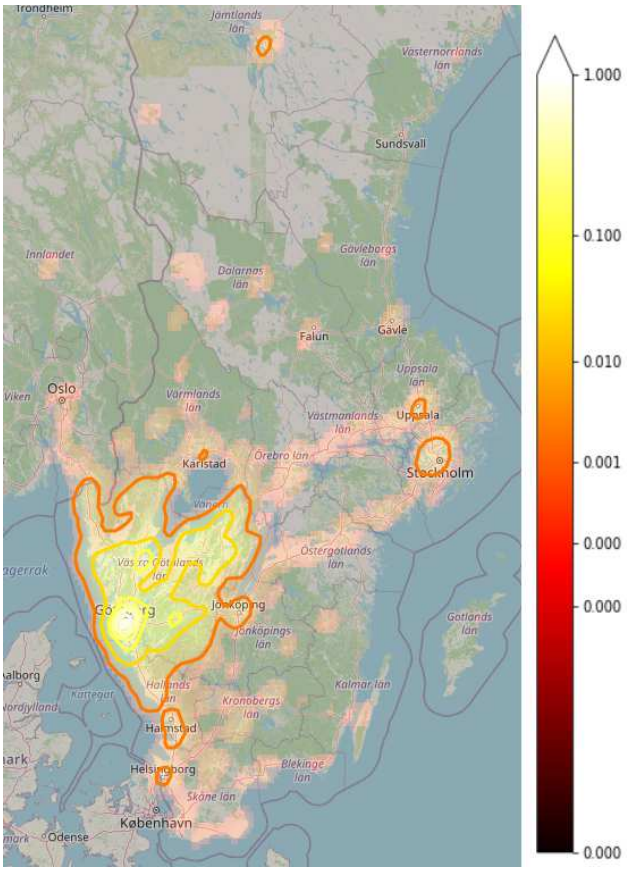


Figure 1: Density map of the trips made by 700 cars in different cities in Sweden. The figure is adapted from [10].

model developed in $PM1$. Furthermore, due to the inequality of the length of the selected sequences of the trips, we use an embedding layer to equalize the length of the state vector for each LSTM layer and feed them to the attention layer. The structure of $PM3$ is shown in Fig. 3 (a). Finally, the last method is the attention-based version of the $PM2$, where two parallel LSTM networks are processing Δt and d separately. Furthermore, the outputs of the LSTM layers go to separate attention layers, after which they are embedded using the embedding layer again. Finally, the outputs of the LSTM and attention layers are fed into a fully connected layer. Fig. 3 (b) illustrates the last method, called $PM4$.

In the following, we describe the fundamental components of these methods, such as LSTM networks, embedding, and attention layer. In addition, we augment the models with an explainability component, as described thereafter.

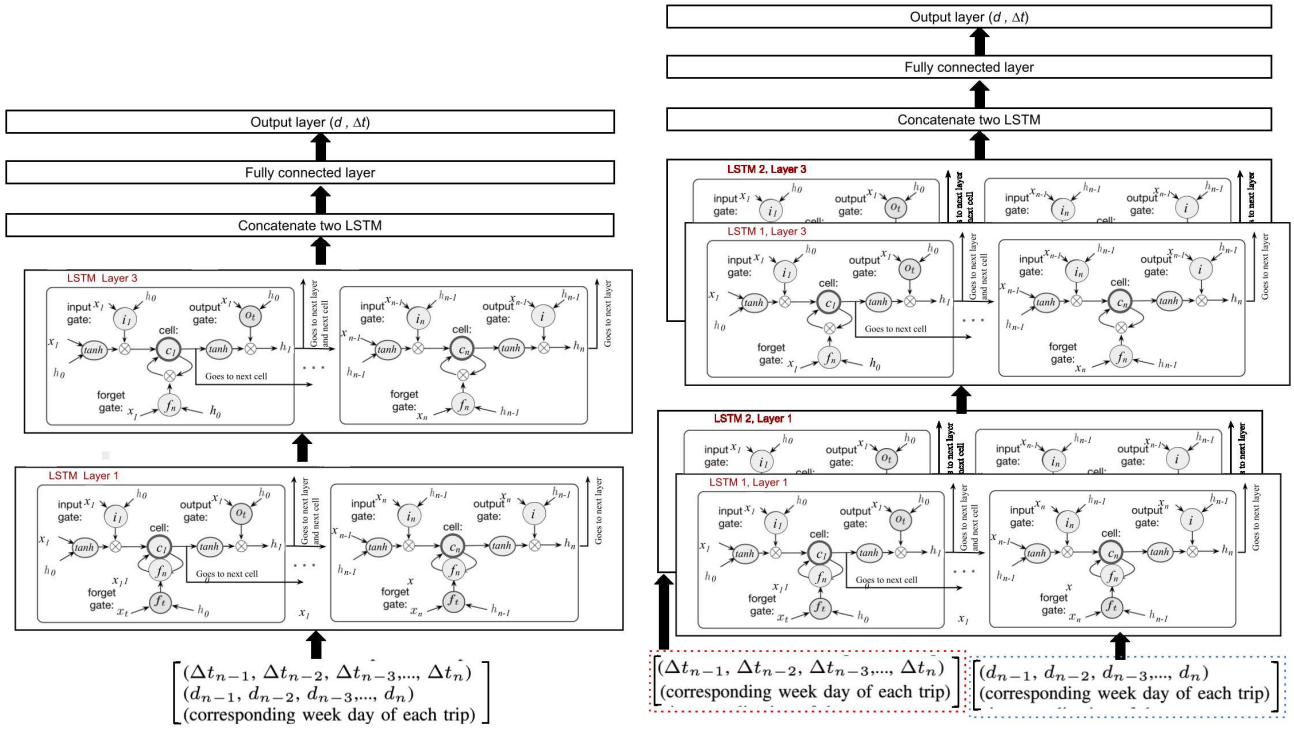
2.2.1 LSTM networks

An LSTM cell, as a stateful operator, is a type of RNN that analyzes the Δt and the distance d of the trips sequentially [14]. For the sake of simplicity, we represent Δt and d, the input vectors to LSTM, together by x_n, where n is the length of the sequence. For each n, x_n is used to compute an output y_n and updates the internal state of the cell at each layer which can be defined as:

$$h_n = f_W(h_{n-1}, x_n) \tag{3}$$

where h_n is a new state and f_W is a nonlinear function parameterized by a set of weights W. h_n implicitly encodes the learned representation of the non-linearity and time-variance of the modeled data. For the LSTM cells, the exact update rules for y_n and h_n are defined in (4)-(6).

$$y = \begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} h_{n-1} \\ x_n \end{bmatrix}, \tag{4}$$



(a) (b)

Figure 2: The unrolled diagram of (a) LSTM and (b) parallel LSTM method. x, h, f, i, c, and o stand for input data, state and forget gate, input gate, cell information, and output gate of LSTM, respectively. W and U stand for weights for each gate.

$$c_n = f \odot c_{n-1} + i \odot g, \tag{5}$$

$$h_n = o \odot \tanh(c_n), \tag{6}$$

where i is the input gate to write information into the cell, f is the *forget gate* that receives the data for resetting the cell state, g determines the magnitude of the influence of the input x_n on the cell and o represents the output gate which controls the information flow to the next layer. σ and $\tanh$ are non-linear activation functions necessary for modelling non-linear data and $\odot$ is element-wise multiplication. All these cells, in combination, provide a powerful tool to learn features and model the behavior of time-variant data in order to be used as a predictor. The parallel version of the LSTM method in the mathematical derivation is illustrated in Algorithm (I).

2.2.2 Attention-based version of LSTM

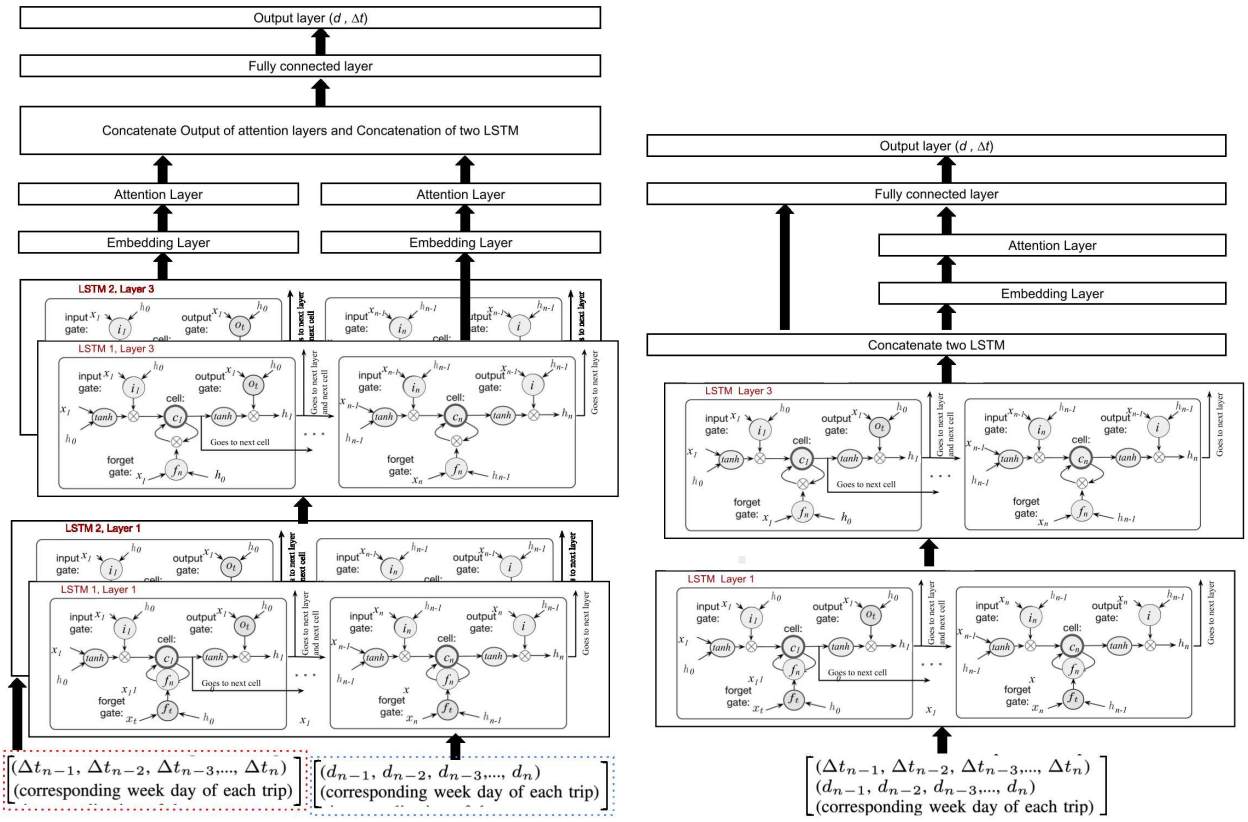
The standard LSTM cannot detect which part of the sequence is relevant for aspect-level forecasting. In order to address this issue, we design an attention-based mechanism that can capture the key part of historical data in response to a given aspect. Fig. 3 (a) represents the architecture of this At-LSTM model.

Let $H \in \mathbb{R}^{k \times n}$ be a matrix consisting of hidden vectors $[h_1, ..., h_n]$ that the LSTM produces, where k is the size of hidden layers and n is the length of the given sentence. Furthermore, v_a represents the embedding of both the aspect and $e_N \in \mathbb{R}^N$ vectors. The attention mechanism produces an attention weight vector α and a weighted hidden representation r, as follows.

$$M = \tanh\left(\begin{bmatrix} W_h H \\ W_v v_a \otimes e_N \end{bmatrix}\right), \tag{7}$$

$$\alpha = \text{sigmoid}\left(w^T M\right), \tag{8}$$

$$r = H\alpha^T, \tag{9}$$



(a) (b)

Figure 3: The unrolled diagram of (a) parallel LSTM and (b) Parallel At-method. x, h, f, i, c, and o stand for input data, state and forget gate, input gate, cell information, and output gate of LSTM. W and U represent the weights.

where $M \in \mathbb{R}^{(k+k_a)\times n}, \alpha \in \mathbb{R}^N, r \in \mathbb{R}^k$ $W_h \in \mathbb{R}^{k\times k}, W_v \in \mathbb{R}^{k_a \times k_a}$ and $w \in \mathbb{R}^{k+k_a}$ are projection parameters. α is a vector consisting of attention weights and r is a weighted representation of h produced by LSTMs. The operator $\otimes$ in $v_a \otimes e_N = [v; v; \ldots; v]$ means that the operator repeatedly concatenates v for n times, where e_n is a column vector with n samples. $W_v v_a \otimes e_n$ is repeating the linearly transformed v_a as many times as there are data points in the selected window of data. The last state representation can be defined as:

$$h^\star = \tanh(W_{p^r} + W_x h_n), \tag{10}$$

where, $h^\star \in R_k, W_p$ and W_x are the weights to be inferred during training. Motivated from [20], we find that this model works practically better if we add $W_x h_n$ into the final representation of the sequence of information.

The attention mechanism helps the model to apprehend the essential part of the information in the selected window of historical data where diverse characteristics are assessed.The parallel version of the LSTM method in the mathematical derivation is illustrated in Algorithm (II)

2.3 Explainability

Here, we develop an explainer that is not only model-agnostic and post-hoc, but also specifically tailored to the At-LSTM networks. To be able to explain the predictor algorithm, the method must provide both trip and attributions of extracted features throughout the selected window of the sequence from historical data. The aim is to provide explainability of the sequential analysis while maintaining three desirable properties: accuracy, missingness, and consistency [21]. Therefore, we use an explainable, model-agnostic At-LSTM that is developed based on the TimeSHAP method introduced in [5]. TimeSHAP defines a *significance* index to the features or trips in the input that specifies the extent to which those features or trips impact the prediction task. To describe a piece of sequential information, $X \in R^{fxD}$, with D trips and f features of each trip, TimeSHAP fits a linear function g (explainer) that matches the local output of a complex explainer g by minimizing the loss function defined in [5].

Algorithm 1: LSTM algorithm :

n : number of trips in selected window of the historical data
d : the distance that a vehicle travels on each trip
k : the number of LSTM networks
Δt : time difference of the trip from the previous trip
Read (Δt_{n-1}, Δt_{n-2}, Δt_{n-3},..., Δt_n)
Read (d_{n-1}, d_{n-2}, d_{n-3},..., d_n)
Read (corresponding weekday of each trip)
Max-min normalization of data
do in parallel
  $k \leftarrow k + 1$
  LSTM Networks 1 (for prediction of Δt_{n+1})
  LSTM Networks 2 (for prediction of d_{n+1})
  while $i <= n$ *(number of neurons in* FC^1*)* **do**
    First LSTM layer:
    $n \leftarrow n + 1$
    $i_n^1 = \sigma_g \left(W_i FC^{1,k} + U_i h_{n-1}^{1,k} + b_i \right)$ ▷ input gate
    $f_n^{1,k} = \sigma_g \left(W_f FC^{1,k} + U_f h_{n-1}^{1,k} + b_f \right)$ ▷ forget gate
    $o_n^{1,k} = \sigma_g \left(W_o FC^{1,k} + U_o h_{n-1}^{1,k} + b_o \right)$ ▷ output gate
    $\tilde{c}_n^{1,k} = \tanh \left(W_c FC^{1,k} + U_c h_{n-1}^{1,k} + b_c \right)$
    $c_n^1 = f_n^1 \circ c_{n-1}^1 + i_n^{1,k} \circ \tilde{c}_n^{1,k}$ ▷ cell information
    $h_n^{1,k} = o_n^{1,k} \circ \tanh \left(c_n^{1,k} \right)$ ▷ state goes to next layer
    •
    •
    •
    3, k[rd] LSTM layer:
    $i_n^{3,k} = \sigma_g \left(U_i h_{n-1}^{3,k} + b_i \right)$ ▷ input gate
    $f_n^{3,k} = \sigma_g \left(U_f h_{n-1}^{3,k} + b_f \right)$ ▷ forget gate
    $o_n^{3,k} = \sigma_g \left(U_o h_{n-1}^{3,k} + b_o \right)$ ▷ output gate
    $\tilde{c}_n^{3,k} = \tanh \left(U_c h_{n-1}^{3,k} + b_c \right)$
    $c_n^{3,k} = f_n^{3,k} \circ c_{n-1}^{3,k} + i_n^{3,k} \circ \tilde{c}_n^{3,k}$ ▷ cell information
    $h_n^{3,k} = o_n^{3,k} \circ \tanh \left(c_n^{3,k} \right)$ ▷ state goes to FC layer
  end
end
Concatenate the result of two LSTM networks :

$$CL = Concatenate. \begin{bmatrix} h_1^{3,1} & h_2^{3,1} & \dots & h_n^{3,1} \\ h_1^{3,2} & h_2^{3,2} & \dots & h_n^{3,2} \end{bmatrix}$$

First FC layer:
$FC^1 = W_F.(CL)$
Second FC layer:
$[\Delta t_{n+1}, d_{n+1}] = W_F.(FC^1)$

The function g that approximates $f\left(h_X(z)\right)$ is denied as

$$f\left(h_X(z)\right) \approx g(z) = b_0 + \sum_{i=1}^{m} w_i z_i, \tag{11}$$

where the term $b_0 = f\left(h_X(\mathbf{0})\right)$ is a bias value and resembles the output model with trips and features ticked off (dubbed base score). The set $\{w_i, i \in \{1, \dots, m\}\}$ refers to the weights representing the impact of each trip/feature ($m = f$ or $m = D$), for each dimension that is going to be explained. The perturbation function $h_X : \{0,1\}^m \mapsto \mathbb{R}^{d \times l}$ maps a coalition $z \in \{0,1\}^m$ to the original input space $x \in R^{f \times D}$. Note that the sum of significance indices for all the features corresponds to the difference between the score of the model $f(X) = f\left(h_X(\mathbf{1})\right)$ and the score which is $h_X(\mathbf{0})$.

3 Experimental Results

In this section, we investigate the different models and compare their performances on our real-world dataset. We perform grid-search-based hyperparameter tuning for all methods to set their hyperparameters.

Algorithm 2: LSTM algorithm :

n : number of trips in selected window of the historical data
d : the distance that a vehicle travels on each trip
k : the number of LSTM networks
Δt : time difference of the trip from the previous trip
Read (Δt_{n-1}, Δt_{n-2}, Δt_{n-3},..., Δt_n)
Read (d_{n-1}, d_{n-2}, d_{n-3},..., d_n)
Read (corresponding weekday of each trip)
Max-min normalization of data
do in parallel
  $k \leftarrow k + 1$
  LSTM Networks 1 (for prediction of Δt_{n+1})
  LSTM Networks 2 (for prediction of d_{n+1})
  while $i <= n$ *(number of neurons in* FC^1*)* **do**
    First LSTM layer:
    $n \leftarrow n + 1$
    $i_n^1 = \sigma_g \left(W_i FC^{1,k} + U_i h_{n-1}^{1,k} + b_i \right)$ ▷ input gate
    $f_n^{1,k} = \sigma_g \left(W_f FC^{1,k} + U_f h_{n-1}^{1,k} + b_f \right)$ ▷ forget gate
    $o_n^{1,k} = \sigma_g \left(W_o FC^{1,k} + U_o h_{n-1}^{1,k} + b_o \right)$ ▷ output gate
    $\tilde{c}_n^{1,k} = \tanh \left(W_c FC^{1,k} + U_c h_{n-1}^{1,k} + b_c \right)$
    $c_n^1 = f_n^1 \circ c_{n-1}^1 + i_n^{1,k} \circ \tilde{c}_n^{1,k}$ ▷ cell information
    $h_n^{1,k} = o_n^{1,k} \circ \tanh \left(c_n^{1,k} \right)$ ▷ state goes to next layer
    •
    •
    •
    3, k[rd] LSTM layer:
    $i_n^{3,k} = \sigma_g \left(U_i h_{n-1}^{3,k} + b_i \right)$ ▷ input gate
    $f_n^{3,k} = \sigma_g \left(U_f h_{n-1}^{3,k} + b_f \right)$ ▷ forget gate
    $o_n^{3,k} = \sigma_g \left(U_o h_{n-1}^{3,k} + b_o \right)$ ▷ output gate
    $\tilde{c}_n^{3,k} = \tanh \left(U_c h_{n-1}^{3,k} + b_c \right)$
    $c_n^{3,k} = f_n^{3,k} \circ c_{n-1}^{3,k} + i_n^{3,k} \circ \tilde{c}_n^{3,k}$ ▷ cell information
    $h_n^{3,k} = o_n^{3,k} \circ \tanh \left(c_n^{3,k} \right)$

$$V_a = embed \begin{bmatrix} h_1^{1,k} & h_2^{1,k} & \dots & h_n^{1,k} \\ h_1^{2,k} & h_2^{2,k} & \dots & h_n^{2,k} \\ h_1^{3,k} & h_2^{3,k} & \dots & h_n^{3,k} \end{bmatrix}$$

    $\alpha = sigmoid.(W_F.\, v_a)$ ▷ state goes to concatenation layer
  end
end
Concatenate the result of two LSTM networks :

$$CL = Concatenate. \begin{bmatrix} Attention^1 & & & \\ Attention^2 & & & \\ h_1^{3,1} & h_2^{3,1} & \dots & h_n^{3,1} \\ h_1^{3,2} & h_2^{3,2} & \dots & h_n^{3,2} \end{bmatrix}$$

First FC layer:
$FC^1 = Relu(W_F.(CL))$
Second FC layer:
$[\Delta t_{n+1}, d_{n+1}] = sigmoid(W_F.(FC^1))$

Table 1: Investigation of the effect of the hyperparameter tuning and their results for the proposed methods. Abbreviations used on this table: parameters (P), methods(M), error (err.), variation (Var.), average (Avg.) learning rate (lr), Mean Absolute Error (MAE), Mean Squared Error (MSE), Log Hyperbolic Cosine (LHC), Huber loss (HL), Mean Squared Logarithmic Error (MSLE), Poisson loss (PS)

M \ P	hyperparameters	best value	P.err. Var.(%)
LSTM	window size (1:1:14 days)	5	23.8
	LSTM layers (1:1:5)	3	18
	# neurons in LSTM layers (20:10:150)	60,120,60	19
	FL layers (1:1:3)	2	3
	batch size (16:16:512)	128	5.2
	lr (0.00001:0.05:0.1)	0.01	17
	optimizer (SGD, Adam, Adagrad, RMSProp)	Adam	6.5
	Loss (MAE, MSE, LHC, HL, MSLE, PS)	MAE	7.3
Parallel LSTM	window size (1:1:14 days)	8	25.8
	LSTM layers (1:1:5)	3	15
	# neurons in LSTM layers (20:10:150)	40,60,40	10
	FL layers (1:1:3)	2	3
	batch size (16:16:512)	128	6
	lr (0.00001:0.05:0.1)	0.01	19
	optimizer (SGD, Adam, Adagrad, RMSProp)	Adam	5.3
	Loss (MAE, MSE, LHC, HL, MSLE, PS)	MAE	10.3
Attention based LSTM	window size (1:1:14 days)	8	25.8
	LSTM layers (1:1:5)	3	15
	# neurons in LSTM layers (20:10:150)	60,90,60	23
	# neurons in LSTM attention (4:4:256)	64	21
	FL layers (1:1:3)	2	3
	batch size (16:16:512)	128	5
	lr (0.00001:0.05:0.1)	0.01	17
	optimizer (SGD, Adam, Adagrad, RMSProp)	Adam	6.2
	Loss (MAE, MSE, LHC, HL, MSLE, PS)	MAE	12.3
Parallel Attention based LSTM	window size (1:1:14 days)	8	25.8
	LSTM layers (1:1:5)	3	15
	# neurons in LSTM layers (20:10:150)	40,60,40	10
	# neurons in LSTM attention (4:4:256)	64	10
	FL layers (1:1:3)	2	3
	batch size (16:16:512)	128	5
	lr (0.00001:0.05:0.1)	0.01	17
	optimizer (SGD, Adam, Adagrad, RMSProp)	Adam	6.2
	Loss (MAE, MSE, LHC, HL, MSLE, PS)	MAE	12.3

Table 2: Investigation of proposed methods and their results for the two training fashions. Abbreviations used on this table: prediction (Pred.), error (Err.), validation (Val.)

Scenario \ Methods	single LSTM	Parallel LSTM	At-LSTM	Parallel At-LSTM
Pred. Err. (%) for 70 (%) Training 15 (%) Val. 15 (%) Test	26.91	12.76	13.32	4.31
Pred. Err. (%) for Cross Val.	26.01	10.97	12.49	3.99

Considering the training time for each hyperparameter choice, we have selected 25 % of data divided into test and training parts. The hyperparameters are aimed tuned, where the range of the search, steps on each search, and the corresponding results are shown in Table 1. As can be seen, the choice of the window of data, LSTM structure in all methods, and learning rate have the highest variation effect on the prediction error.

Note that the evaluation of the performance of the proposed methods is based on the prediction error we obtained using, defined as:

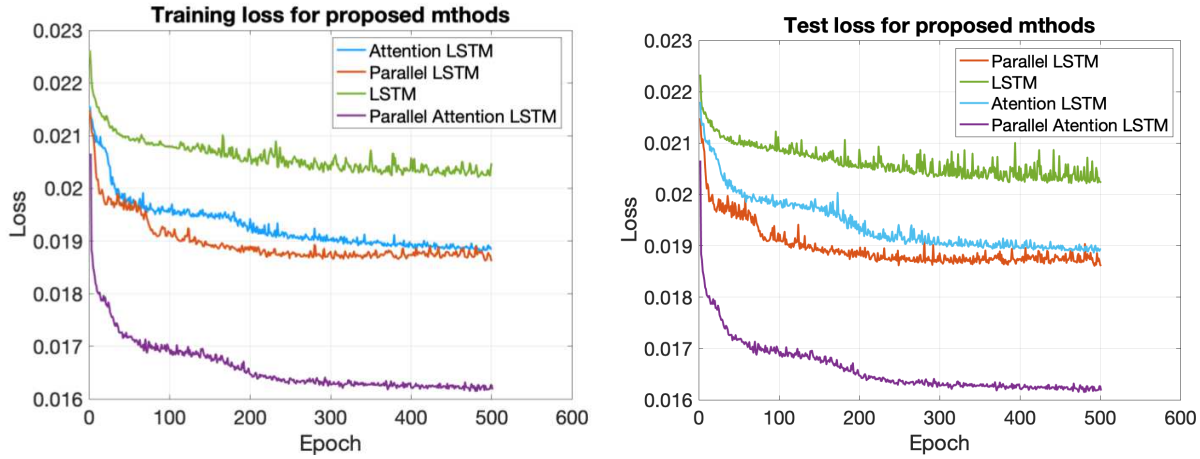
$$100 \cdot \sqrt{\frac{\sum_{n=1}^{N} \left| X[n] - \hat{X}[n] \right|^2}{\sum_{n=1}^{N} |X[n]|^2}}, \tag{12}$$

where $X[n]$ is the true value (label) and $\hat{X}[n]$ is the predicted value.

We perform our experiments *Python* environment using a workstation with an Intel i7 3.40 GHz CPU, 48GB RAM, and two NVIDIA GeForce RTX 4080 GPUs.

3.1 Comparison of different methods

Once all the hyperparameters are tuned, we train the networks in two ways. In the first scenario, we divide the data into three subsets: 70% for training, 15% for validation, and 15% for testing. In the second scenario, rather than having a fixed dataset for training and validation, we use cross-validation to train the neural network on each method. We randomly choose 75 % of the data for training and 15 % for 10 times. At each training, the weights of the previously trained networks will be updated when seeing a new dataset. Note that the test dataset is never used in the cross-validation process and is the same dataset used as test data for the first case scenario. We observe that cross-validation-based transfer learning improved the prediction accuracy for all methods (see Table 2). The convergence of the learning procedure is illustrated in Fig. 4 indicates the prediction error on the training and test dataset per epoch. As an example of the performance of the proposed methods, we illustrate the prediction of the distance of the trips on the part of the test dataset in Fig. 5. To avoid cluttering the results, only the LSTM and parallel At-LSTM method predictions are illustrated where they yield the highest and lowest prediction error, respectively. The blue line is the true labels, the red dashed line is the prediction using parallel At-LSTM, and the yellow dotted line is the prediction of true labels by the LSTM method. Observing the prediction results in Fig. 5, we conclude that parallel At-LSTM indeed performs better in the case of seldom trip patterns, whereas, in the repetitive patterns, pure LSTM yields a similar quality. Thus, this can be why the prediction accuracy of parallel At-LSTM is higher than the other methods developed in this paper.



(a) (b)

Figure 4: prediction error on the (a) training phase and (b) test data monitored during the learning process

3.2 Explainability

As described before, timeSHAP provides local explanations for the model's behavior regarding a prediction. These descriptions can help improve the model in several aspects, such as auditing or model refining. Nevertheless, end-users, the fraud analysts, can mainly utilize the timeSHAP results to assist their decision-making. It can also be used to gather information and KPIs from end users to develop more sophisticated methods, such as online learning. The result of using the TimeSHAP method for the explainability of the performance of the networks can be seen in Fig. 6. We select two sequences: (i) we observe the trip d from 1-49 and predict the trip number 50, and (ii) we observe trips 5-54 and predict the tip number 55. We extract timeSHAP values for historical trips in both scenarios. Doing

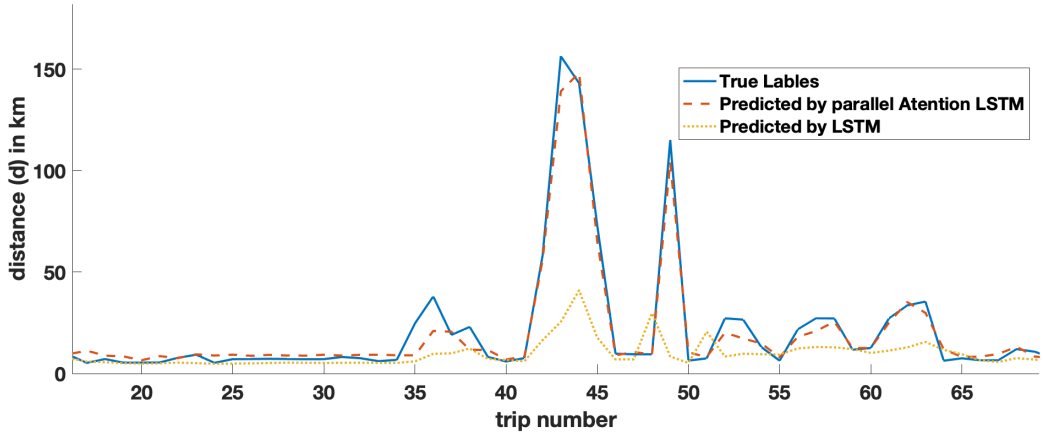


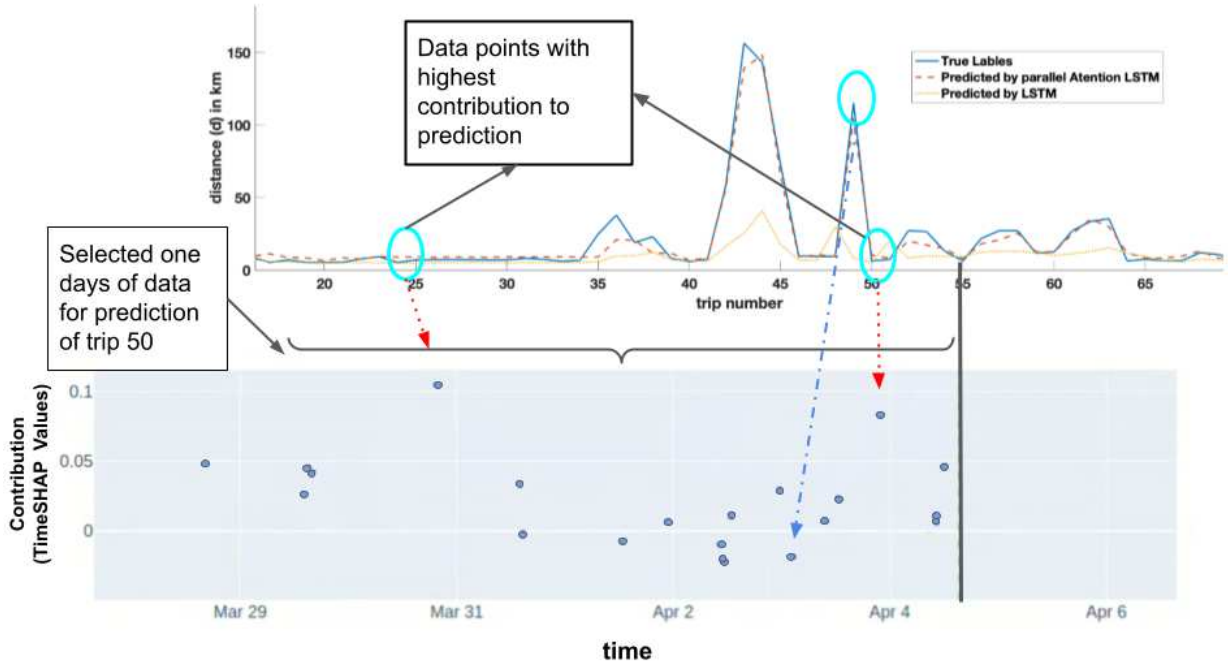
Figure 5: Prediction of the distance of the trips on the part of the test dataset. The blue line is the true labels, the red dashed line is the prediction using parallel At-LSTM, and the yellow dotted line is the prediction of true labels by the LSTM method.

such, indicates the contribution of each trip in the prediction task. Therefore, we can observe the performance of the proposed methods on how well they have learned the relevant patterns in the dataset for an accurate or inaccurate prediction. Considering the fact that the output of all methods is almost the same for scenario (i), to avoid cluttering the results, we only illustrate the timeSHAP values for the parallel At-LSTM method (see Fig. 6 (a)). As can be seen from the prediction of the distance of trip number 55, which has a true distance of 5.3 km, the information of trips 51 and 19 (pointed out with red-dotted arrows) have the highest timeSHAP values. Whereas the trips with distances above 50 km have negative timeSHAP value (see blue dot-dashed arrow). However, in scenario (ii), the timeSHAP values differ as prediction accuracy is significantly low for the LSTM method compared to parallel At-LSTM. Observing the timeSHAP values for both methods in Fig. 6 (b), the trips around 42-45, which are the longest in the distance, have the highest contribution to the prediction of trip 49. On the other hand, the trips with low distances have the least contribution to the prediction. On the other hand, observing the extracted timeSHAP values for the LSTM method in scenario (ii), the trips with the highest distance have a low contribution, and trips with a low distance have the highest contribution. This can be a reason why the prediction accuracy of LSTM is lower compared to the parallel At-LSTM method.

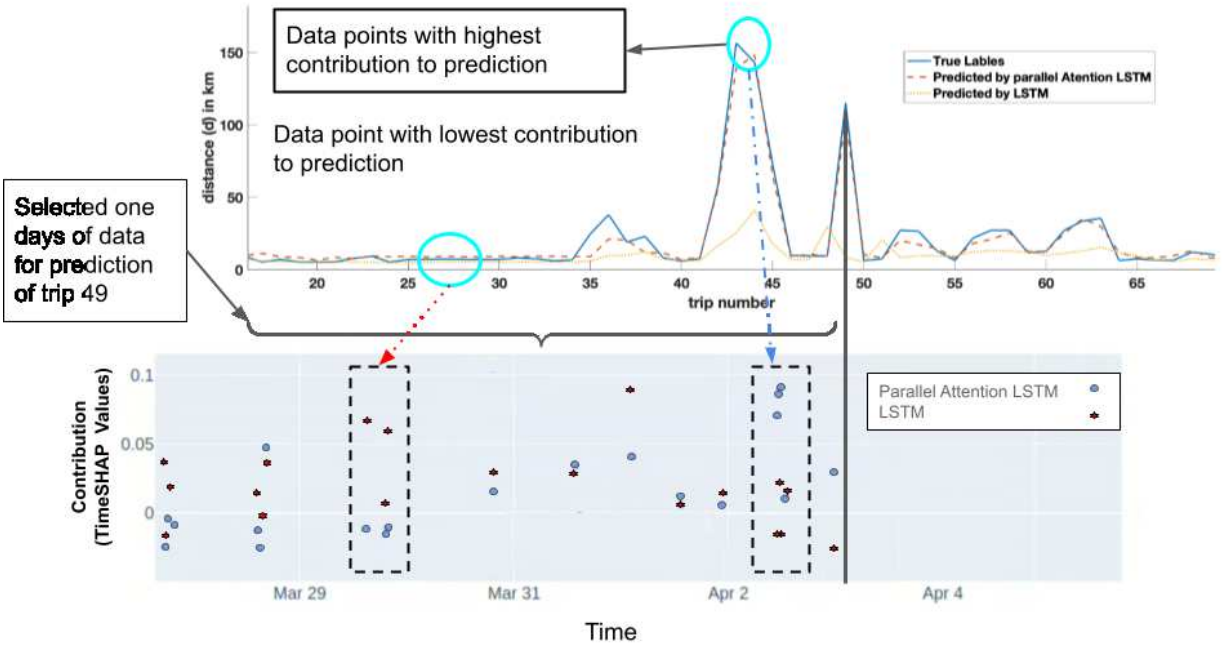
Thereby, we observe that the explainability method is able to successfully suggest the reasoning behind an accurate prediction. Furthermore, we can also see that the parallel At-LSTM has learned the historical patterns for predicting the future trip distance (d). Finally, it may be concluded that the statement "the standard LSTM cannot detect which is the sequence part for aspect-level sentiment forecasting" is true in this setup.

4 CONCLUSION

In this paper, we proposed a novel deep learning framework for predicting the future distance and time of vehicle trips. We showed that the parallel attention-based LSTMs yield minimal prediction error compared to using only LSTM, attention-based LSTM, and two parallel LSTMs. This method outperformed the alternatives in terms of robustness and learning complicated patterns. Moreover, we developed a timeSHAP-based explainability method to show how the patterns have been learned by a method. Also, we observed that tuning the parameters and hyperparameters, such as choosing the number of days, features for creating sequences, or the learning rate of the optimization algorithms, can affect the prediction accuracy. Thus, it can be concluded that for an accurate predictor model, four important steps need to be followed: the art of feature selection, feature engineering, innovative selection and structuring of neural networks, and hyperparameter optimization of the machine learning methods. In future work, we are planning to create vehicle-specific models where the result of such can create a mother and universal model by using federated learning approaches.



(a)



(b)

Figure 6: The Estimated timeSHAP values for the prediction of (a) trip 50 and (b) trip 49 in Fig. 5 using seven days of data.

References

[1] Influence of plug-in hybrid electric vehicle charging strategies on charging and battery degradation costs. *Energy Policy*, 46:511–519, 2012.

[2] Niklas Åkerblom, Yuxin Chen, and Morteza Haghir Chehreghani. An online learning framework for energy-efficient navigation of electric vehicles. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, pages 2051–2057, 2020.

[3] Niklas Åkerblom, Yuxin Chen, and Morteza Haghir Chehreghani. Online learning of energy consumption for navigation of electric vehicles. *Artificial Intelligence*, 317:103879, 2023.

[4] Niklas Åkerblom, Fazeleh Sadat Hoseini, and Morteza Haghir Chehreghani. Online learning of network bottlenecks via minimax paths. *Machine Learning*, 112(1):131–150, 2023.

[5] João Bento, Pedro Saleiro, André F Cruz, Mário AT Figueiredo, and Pedro Bizarro. Timeshap: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2565–2573, 2021.

[6] Malte Bieler, Anders Skretting, Philippe Büdinger, and Tor-Morten Grønli. Survey of automated fare collection solutions in public transportation. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[7] Yuxin Chen and Morteza Haghir Chehreghani. Trip prediction by leveraging trip histories from neighboring users. In *25th IEEE International Conference on Intelligent Transportation Systems, ITSC*, pages 967–973. IEEE, 2022.

[8] Boris Chidlovskii. Improved trip planning by learning from travelers' choices. In *MUD@ ICML*, 2015.

[9] Hugo Neves de Melo, João Pedro F. Trovão, Paulo G. Pereirinha, Humberto M. Jorge, and Carlos Henggeler Antunes. A controllable bidirectional battery charger for electric vehicles with vehicle-to-grid capability. *IEEE Transactions on Vehicular Technology*, 67(1):114–123, 2018.

[10] Victor Eberstein, Jonas Sjöblom, Nikolce Murgovski, and Morteza Haghir Chehreghani. A unified framework for online trip destination prediction. *Machine Learning*, 111(10):3839–3865, 2022.

[11] Mehrdad Ehsani, Milad Falahi, and Saeed Lotfifard. Vehicle to grid services: Potential and applications. *Energies*, 5(10):4076–4090, 2012.

[12] Jonathan P Epperlein, Julien Monteil, Mingming Liu, Yingqi Gu, Sergiy Zhuk, and Robert Shorten. Bayesian classifier for route prediction with markov chains. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 677–682. IEEE, 2018.

[13] Alireza Ermagun, Yingling Fan, Julian Wolfson, Gediminas Adomavicius, and Kirti Das. Real-time trip purpose prediction using online location-based search and discovery services. *Transportation Research Part C: Emerging Technologies*, 77:96–112, 2017.

[14] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[15] Sten Karlsson. The swedish car movement data project final report. Technical report, Chalmers University of Technology, 2013.

[16] Soohwan Kim, Hoyoung Jeong, and Hoseong Lee. Cold-start performance investigation of fuel cell electric vehicles with heat pump-assisted thermal management systems. *Energy*, 232:121001, 2021.

[17] Haitao Min, Zhaopu Zhang, Weiyi Sun, Zhaoxiang Min, Yuanbin Yu, and Boshi Wang. A thermal management system control strategy for electric vehicles under low-temperature driving conditions considering battery lifetime. *Applied Thermal Engineering*, 181:115944, 2020.

[18] Paul Nelson, Dennis Dees, Khalil Amine, and Gary Henriksen. Modeling thermal management of lithium-ion pngv batteries. *Journal of Power Sources*, 110(2):349–356, 2002.

[19] Ghazaleh Panahandeh. Driver route and destination prediction. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 895–900. IEEE, 2017.

[20] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[21] LLOYD SHAPLEY. A value for n-person games. contributions to the theory of games ii, kuhn, h., tucker, a, 1953.

[22] Ayaka Shigemoto, Takuma Higo, Yuki Narita, Seiji Yamazoe, Toru Uenishi, and Yasushi Sekine. Elucidation of catalytic no x reduction mechanism in an electric field at low temperatures. *Catalysis Science & Technology*, 2022.

[23] Matthew Shirk and Jeffrey Wishart. Effects of electric vehicle fast charging on battery life and vehicle performance. *SAE Technical Paper 2015-01-1190*, 4 2015.

[24] Chunjie Zhou, Pengfei Dai, and Zhenxing Zhang. Orientsts-spatio temporal sequence searching for trip planning. *International Journal of Web Services Research (IJWSR)*, 15(2):21–46, 2018.