

Article

Energy Management Strategy Based on a Novel Speed Prediction Method

Jiaming Xing¹, Liang Chu¹, Zhuoran Hou¹, Wen Sun²  and Yuanjian Zhang^{3,*}

¹ State Key Laboratory of Automotive Dynamic Simulation and Control, Jilin University, Changchun 130021, China; xingjm19@mails.jlu.edu.cn (J.X.); chuliang@jlu.edu.cn (L.C.); houzr20@mails.jlu.edu.cn (Z.H.)

² College of Automotive Engineering, Changzhou Institute of Technology, Changzhou 213032, China; sunw@czu.cn

³ School of Mechanical and Aerospace Engineering, Queen's University Belfast, Belfast BT9 5AG, UK

* Correspondence: Y.Zhang@qub.ac.uk

Abstract: Vehicle speed prediction can obtain the future driving status of a vehicle in advance, which helps to make better decisions for energy management strategies. We propose a novel deep learning neural network architecture for vehicle speed prediction, called VSNet, by combining convolutional neural network (CNN) and long-short term memory network (LSTM). VSNet adopts a fake image composed of 15 vehicle signals in the past 15 s as model input to predict the vehicle speed in the next 5 s. Different from the traditional series or parallel structure, VSNet is structured with CNN and LSTM in series and then in parallel with two other CNNs of different convolutional kernel sizes. The unique architecture allows for better fitting of highly nonlinear relationships. The prediction performance of VSNet is first examined. The prediction results show a RMSE range of 0.519–2.681 and a R2 range of 0.997–0.929 for the future 5 s. Finally, an energy management strategy combined with VSNet and model predictive control (MPC) is simulated. The equivalent fuel consumption of the simulation increases by only 4.74% compared with DP-based energy management strategy and decreased by 2.82% compared with the speed prediction method with low accuracy.

Keywords: speed prediction; deep learning; energy management strategy; model predictive control



Citation: Xing, J.; Chu, L.; Hou, Z.; Sun, W.; Zhang, Y. Energy Management Strategy Based on a Novel Speed Prediction Method. *Sensors* **2021**, *21*, 8273. <https://doi.org/10.3390/s21248273>

Academic Editor: Felipe Jiménez

Received: 17 November 2021

Accepted: 9 December 2021

Published: 10 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the development of technology, speed prediction has been widely used in economic, industry and transportation [1–6]. In the vehicle industry, speed prediction has become an important part of intelligent vehicle energy management strategies [7]. As one of the innovative control architectures, hierarchical control architectures equipped with prediction modules have an indispensable contribution in modern control theory [8,9]. In the hierarchical control architecture of new energy vehicles (NEVs), the upper control architecture is mainly responsible for the acquisition of prior information. The vehicle speed prediction, as an effective method to obtain information about future driving conditions, has a direct impact on the control effect of the bottom control [4]. Therefore, the accuracy and efficiency of prediction affect the performance and practicality of the energy management system. Improving the accuracy of vehicle speed prediction will be the main research content of this paper.

The accuracy of speed prediction affects the optimization effect of energy management strategies. In the terms of instantaneous-optimization-based energy management strategies, Jiazheng PEI used six different speed prediction methods to construct model prediction control (MPC) energy management strategies, respectively. The results showed that the method with smaller speed prediction error obtained better fuel economy [10]. In the terms of global-optimization-based energy management strategies, dynamic programming (DP), which belongs to the global-optimization-based energy management strategies, requires prior information of the vehicle driving conditions before optimizing the optimal control

scheme [11]. Prior information and heavy computation pose difficulties for its application in real-time control. To remedy this deficiency, Menglin Li combined speed prediction and DP. The driving cycle was divided into finite intervals and the speed prediction method of deep neural network was used to solve in segments within intervals. The results showed that the speed prediction accuracy could be improved to reduce the fuel consumption by 3.34% per 100 km [12]. The higher the accuracy of speed prediction, the better the optimization effect.

One of the challenge in studying energy management strategies for hybrid electric vehicle (HEV) is improving the accuracy of vehicle speed prediction [13]. With the development of speed prediction algorithms, more and more methods are available. The commonly used methods for speed prediction are linear regression analysis [14,15], vehicle dynamics model prediction [16,17], stochastic prediction [18,19] and machine learning [20–23].

Linear regression analysis is a statistical analysis method to determine the quantitative relationship between two or more variables [24]. Based on the availability of extensive observational data, mathematical and statistical methods are used to establish the regression relationship between vehicle speed and its influence factors. Mirbaha, B et al. selected traffic and weather data of a two-lane highway located in northwestern Iran for their analysis. A linear regression model was used for speed prediction and analysis of the correlation between the average vehicle speed and variables including vehicle flow, percentage of heavy vehicles, rainfall, etc. [14]. Maji, A et al. used a multiple linear stepwise regression method to develop 85th percentile and 98th percentile speed prediction models for four-lane horizontal curves on rural roads and obtained a minimum Root Mean Square Error (RMSE) of 4.37 km/h [15]. The regression analysis method has a simple model with poor robustness, which cannot fit highly nonlinear relationships well.

The vehicle dynamics model prediction method is based on a vehicle dynamics model that calculates the vehicle acceleration by analyzing the wheel forces to evaluate the vehicle speed. Li et al. built a vehicle controller by predicting the maximum vehicle speed based on the vehicle lateral dynamics when entering a curve. The controller provided an energy management strategy that improved the safety and fuel economy compared with equivalent consumption minimization strategy [16]. Ma, Z et al. proposed a novel method to identify the normal load of each tire in a heavy truck in order to estimate the longitudinal velocity. The simulation showed that the predicted velocity could follow the trend of the real velocity and no significant lag was found [17]. Since the vehicle dynamics model needs to consider various parameters such as turning radius, vehicle mass, tire stiffness, etc., the model is simplified compared with the actual vehicle in order to reduce the modeling difficulty. The speed prediction accuracy is limited by the model accuracy and simplification conditions.

As an important method for stochastic prediction, Markov chains are widely used [25,26]. Markov chain is a stochastic data-driven model that utilizes the state transfer matrix and the current state to predict the future driving state. Jaewook shin et al. proposed a vehicle speed prediction algorithm based on a Markov chain model with speed constraints. The algorithm was experimentally verified to provide a mean square error of 3.8041 km/h [18]. Karbowski et al. used Markov chains to predict the future speed of the vehicle and designed an energy management strategy for plug-in hybrid vehicles. Fuel savings of up to 5.7% were achieved on a 36 km route in Munich [19]. The Markov chain method captures vehicle speed to construct a state transfer matrix for speed prediction. The speed variation is more in line with the real driving conditions of the vehicle. However, little historical information is considered and stochastic prediction leads to large fluctuations in vehicle speed.

The methods described above have simpler prediction models with fewer adjustable parameters, which means weaker fitting of nonlinear relationships and poorer accuracy in predicting vehicle speeds. The development of machine learning techniques has brought new solutions for vehicle speed prediction. Machine learning can learn features from a limited amount of observed data, utilize the features to predict output and has a better nonlinear fitting capability [20,27,28]. Machine learning is divided into shallow learning

and deep learning. Traditional shallow learning is weak in feature extraction and requires effort for data processing. If the features are not properly selected, there is no guarantee that there is a stable mapping relationship between the input and output of the sample so that the prediction effect is seriously affected [29,30]. Deep learning differs from traditional shallow learning in that: (1) it emphasizes the depth of the model architecture and improves the nonlinear fitting ability by setting a large number of parameters; (2) it explicitly highlights the importance of feature processing by transforming the feature in the original space to a new feature space through layer-by-layer feature transformation, which makes classification or prediction easier and does not depend on expert experience. In recent years, convolutional neural network (CNN), a powerful deep learning method, has been widely used in prediction due to its excellent feature extraction capability [21,31,32]. Loaiza, F.A. et al. presented a CNN method for speed prediction of a large-scale traffic network. The empirical results showed that the method reduced the convergence time without losing the performance of the predictions [33]. Ma, X. et al. proposed a CNN method that learns traffic as images and predicts large-scale, network-wide traffic speed. The results showed that the CNN outperformed other algorithms on testing data with an average accuracy improvement of 42.91% [34]. CNN reduces the number of parameters to be adjusted by neural networks through perceptual field and weight sharing, minimizes the preprocessing requirements of data and achieves high accuracy vehicle speed prediction. However, CNN is not designed to handle temporal data specifically. For vehicle speeds with time-series characteristics, the prediction accuracy of CNN is subject to loss.

Compared with CNN, long short-term memory (LSTM) neural network with long sequence processing capability has good performance in sequence prediction to detect long-term dependencies [35]. Hochreiter and Schmidhuber proposed LSTM in 1997 [36]. It has been refined by many researchers in the following decade [37–39]. In the field of transportation, LSTM is one of the most commonly used prediction models. Ma et al. pioneered the introduction of LSTM for vehicle speed prediction and found that the LSTM neural network achieved the best results in terms of prediction accuracy and stability by comparing it with other neural networks [22]. Yeon, K. et al. used information from the past 30 s to predict the future vehicle speed based on LSTM. The results showed that LSTM has better prediction performance compared with CNN [23]. LSTM gets an edge on sequence problem and has long time memory capability. However, except for the temporal features, it has a poor ability to extract the remaining features.

The complexity of machine learning allows for a stronger nonlinear fitting capability. A high accuracy model for speed prediction can be obtained by training a large amount of data. However, a general machine learning model that solves a series of complex problems does not exist. Individual model always has certain limitations in application.

In the context, this paper combines CNN and LSTM to propose a novel neural network structure based deep learning specifically for vehicle speed prediction, named VSNet. VSNet can identify the mapping relationship between vehicle signals and vehicle speed to accurately predict the future vehicle speed. In this paper, VSNet is compared with Markov chain combined with Monte Carlo (MCMC), support vector machine (SVM) and CNN to verify the effectiveness of VSNet in predicting vehicle speed. Finally, for a parallel HEV, the four speed prediction methods are combined with MPC for simulation experiments to verify the impact of prediction accuracy on MPC.

The innovations of this paper are as follows.

- (1) The temporal vehicle signals are arrayed as fake images, which are adopted as the input sample of VSNet. Three types of convolutional kernels with different sizes are used to extract the temporal features, vehicle signal features and comprehensive features in a single sample, respectively.
- (2) Generate sample sequences based on time series. LSTM is used to extract the sequential features between the sample.
- (3) The combination of CNN and LSTM is used to construct VSNet, which makes up for the defects of the individual model and enhances the learning ability.

The remainder of this paper is structured as follows. Section 2 introduces the deep learning method and proposes VSNet. Section 3 introduces the vehicle model and MPC model. Section 4 shows the speed prediction results and combines MPC with speed prediction methods for vehicle simulation validation. Section 5 shows the conclusion and future outlook.

2. Speed Prediction Method

Road experiments are conducted to collect raw data of a vehicle. By calculating the Pearson correlation coefficient between each signal and vehicle speed, the top 15 signals that have the strongest correlation with vehicle speed are selected for feature extraction. The fake image is obtained by fixed-step interception and the training set is obtained by rolling interception. The prediction model of vehicle speed is obtained by training the parameters offline through error back propagation. When the vehicle is in motion, the vehicle speed can be predicted online in real time by simply inputting the vehicle information into the trained model. The flow chart of speed prediction is shown by Figure 1.

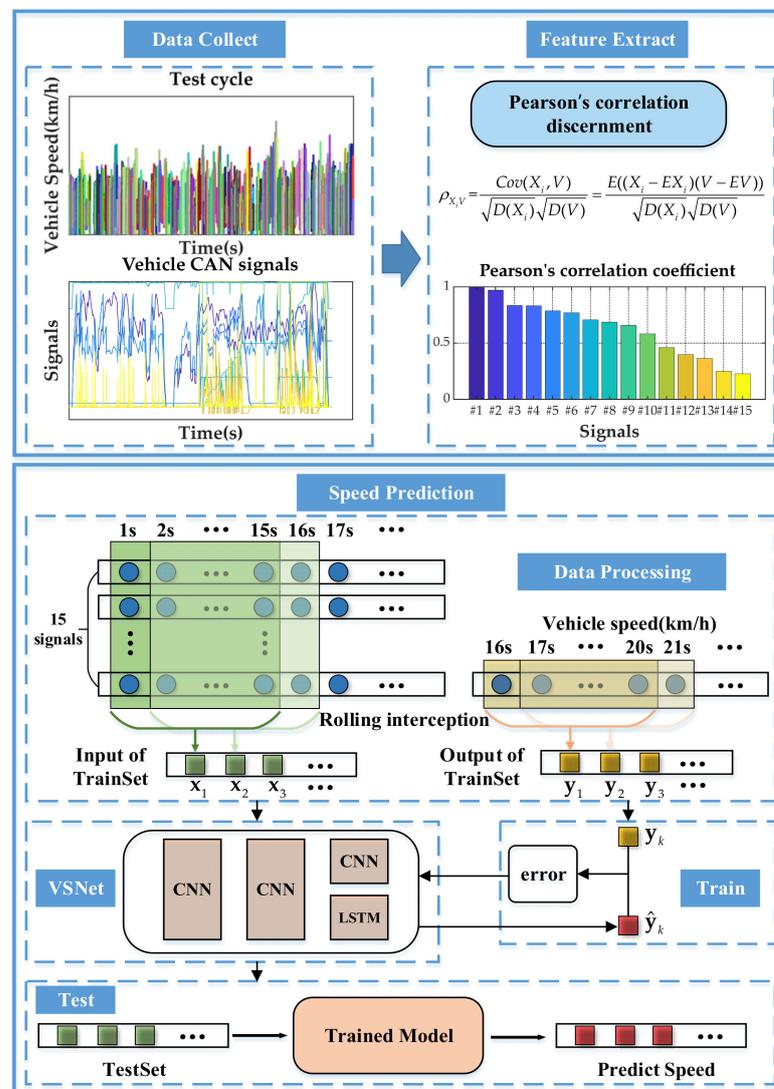


Figure 1. The flow chart of speed prediction.

2.1. Data Processing

In this paper, road experiments are performed on a hybrid electric vehicle to collect data of signals. The data from each signals forms a vector that changes over time. Vectors

need to be arranged into a matrix as a fake image to be input to VSNet. To ensure the reasonableness of fake images, selected signals need to be able to establish a mapping relationship with vehicle speed. In order to make a mapping between input data and predicted vehicle speed, this paper uses Pearson correlation to judge the relationship between each vehicle signal and vehicle speed and selects fifteen signals for subsequent research, which ensures data quality and reduces the input data scale.

Pearson correlation is a method proposed by the British statistician Pearson in the 20th century to calculate the linear correlation [40]. The Pearson correlation coefficient between vehicle signal and vehicle speed can be calculated by

$$\rho_{X_i V} = \frac{Cov(X_i, V)}{\sqrt{D(X_i)}\sqrt{D(V)}} = \frac{E((X_i - EX_i)(V - EV))}{\sqrt{D(X_i)}\sqrt{D(V)}} \quad (1)$$

where E denotes expectation, D denotes variance, Cov denotes covariance, V denotes vehicle speed and X_i denotes the i th vehicle signal. The selected signals and their Pearson correlation coefficients with vehicle speed are shown by Table 1 and Figure 2.

Table 1. Pearson correlation coefficient of selected signals.

CAN Bus Signal	Pearson Correlation Coefficient
Vehicle Speed (km/h)	1
Powertrain Speed (rpm)	0.971832
Engine Speed (rpm)	0.832933
Generator Speed (rpm)	0.83258
Generator Efficiency	0.786406
Motor Speed (rpm)	0.768914
Motor Efficiency	0.707602
Engine Fuel Consumed (gram)	0.686369
Engine Fuel rate (g/kwh)	0.657577
Engine On	0.581088
Clutch On	0.45993
Engine Torque (Nm)	0.397152
Engine Throttle	0.36374
Regen Brake	0.250294
Battery Current (A)	0.225507

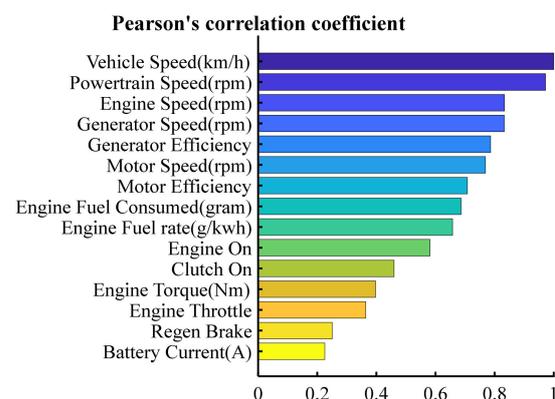


Figure 2. The illustration of Pearson correlation coefficient.

Samples used for training model consists of input and output. The input of samples is obtained by intercepting the selected vehicle signal sequences with sampling time of 15 s and rolling time of 1 s. The output of samples is obtained by intercepting the vehicle

speed sequence with sampling time of 5 s and rolling time of 1 s. The format of samples is described by Equations (2) and (3).

$$x_k = \begin{Bmatrix} S_1^{k-14} & S_1^{k-13} & \dots & S_1^k \\ S_2^{k-14} & S_2^{k-13} & \dots & S_2^k \\ \vdots & \vdots & \ddots & \vdots \\ S_{15}^{k-14} & S_{15}^{k-13} & \dots & S_{15}^k \end{Bmatrix} \quad (2)$$

$$y_k = \begin{Bmatrix} S_1^{k+1} \\ S_1^{k+2} \\ S_1^{k+3} \\ S_1^{k+4} \\ S_1^{k+5} \end{Bmatrix} \quad (3)$$

where k denotes both the time and the sample label; x_k and y_k are input and output of the k th sample, respectively; S_i denotes the i th vehicle signal in Table 1.

2.2. CNN

The data collected in this paper are two-dimensional fake images with vehicle signal dimension and time dimension. Due to the unique structure, CNN has high accuracy regression results and classification results for image processing. Therefore, it is chosen to build speed prediction model. A basic CNN architecture is shown by Figure 3. CNN is a deep feedforward neural network that extracts features of the data within the coverage by a unique convolutional computation [29]. Its structure consists of input layer, intermediate layer and output layer, where the intermediate layer includes convolutional layer, pooling layer, activation function and batch normalization layer.

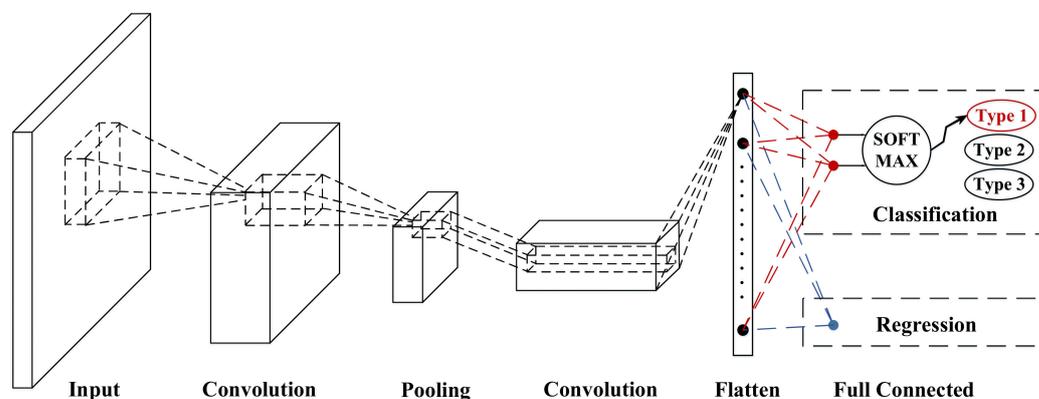


Figure 3. The illustration of a basic CNN.

2.2.1. Convolutional Layer

The role of the convolution layer is compression and purification in order to enhance input signals and reduce noise. The neurons between the convolutional layers are locally connected and share the weights. This design greatly speeds up the training and decreases the number of weights and biases. Local connectivity means that the neurons in the upper layer are connected to the neurons in the lower layer only in a local area, which is also called local receptive field. The mapping from the previous layer to the next layer is called the feature mapping. The weight that defines the feature mapping is called the shared weight. The bias of the feature mapping is called shared bias. Collectively, they are referred to as a convolution kernel. A single convolution kernel can detect only one type of local feature in most cases, so multiple convolution kernels are often needed in order to implement the mapping of multiple features. The rules of convolution operation for multiple input

features are shown by Equation (4). Additionally, for the convolution operation with step size d , the formula is described by Equation (5).

$$Z_{i,j,k}^{l+1} = \sum_{s,m,n} A_{s,j+m-1,k+n-1}^l W_{i,s,m,n}^{l+1} + B_i^{l+1} \quad (4)$$

$$Z_{i,j,k}^{l+1} = \sum_{s,m,n} A_{s,(j-1)d+m,(k-1)d+n}^l W_{i,s,m,n}^{l+1} + B_i^{l+1} \quad (5)$$

where Z denotes the input of the neuron, l denotes the l th layer, i denotes i th feature vector, j and k denotes the j th row and k th column of a feature vector, respectively, A denotes the output of the neuron, s denotes the number of feature vectors in the layer, m and n denotes the value of (m, n) position in a convolution kernel, and B is the bias.

2.2.2. Pooling Layer

The reason why convolution is used to extract features is that features in one region of the input data are most likely to be present in another region as well. Therefore, it is necessary to further aggregate statistics of features at different regions, which is called pooling. The pooling layer is less prone to overfitting and can reduce the feature resolution. The pooling method used in this paper is maximum pooling, whose equation is described by

$$ap_{i,j,k}^l = \max_{m,n} \left\{ a_{i,(j-1)d+m,(k-1)d+n}^l \right\} \quad (6)$$

where ap denotes the output of the pooling layer, d denotes the step size of the pooling, m and n denotes the value of (m, n) position in a pooling kernel, and the rest of the parameters are the same as before.

2.2.3. Activation Function

The activation function is responsible for mapping the input of a neuron to the output. It is crucial for artificial neural network models to learn and understand complex and non-linear functions. The activation function used in this paper is the ELU function, which performs constant operations on positive inputs and exponential nonlinear operations on negative inputs. Its function expression is described by

$$\text{ELU}(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \quad (7)$$

where x is the input of the activation layer and α is the adjustable parameter.

2.2.4. Batch Normalization Layer

The most widely used normalization method in convolutional neural networks is batch normalization. Batch normalization has the role in preventing gradient explosion and gradient disappearance. In the backpropagation process of neural network, the gradient of each layer is calculated by multiplying the gradient passed from the upper layer. If the gradient of each layer is close to 0, the more layers are propagated, the smaller the gradient is. This causes the gradient disappearance during the back propagation. On the contrary, if the gradient of each layer is more than 1, it will lead to gradient explosion. The normalized data is mapped by the activation function to keep the gradient at a suitable level, so that the gradient does not disappear or explode during the training back propagation. In the training process, batch size is used to determine the number of samples selected for once training. Assuming that there are m elements in a batch size, the set $B = \{x_1, \dots, x_m\}$ is formed. The strategy of batch normalization is to first compute the sample mean and

sample variance in turn for B , and then perform normalization, translation and scaling on the sample data. The formulas are described by Equations (8)–(11).

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (8)$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (9)$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \quad (10)$$

$$BN_{\gamma, \beta}(x_i) = \gamma \hat{x}_i + \beta \quad (11)$$

where μ_B is the sample mean, σ_B^2 is the sample variance, ε is an infinitesimal positive number, and γ and β are training parameters.

2.2.5. Output Layer

The CNN implements regression through a fully connected layer. The fully connected layer extracts and integrates all useful information through matrix multiplication, which is equivalent to feature space transformation. Together with the nonlinear mapping of the activation function, multiple layers of fully connected layers can theoretically simulate any nonlinear transformation.

2.3. LSTM

LSTM is a temporal recurrent neural network suitable for processing and predicting important events in time series with relatively long intervals and delays [39]. The structure of LSTM is shown by Figure 4. LSTM adds a unit called cell that judges whether the information is useful or not. Only information that meets the conditions is remembered, the ones that do not are ignored.

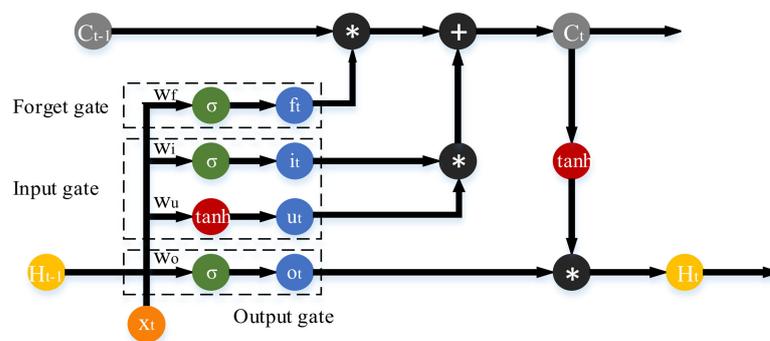


Figure 4. The illustration of a basic CNN.

The sigmoid function is applied as a gate to return a value in the interval from 0 to 1. In this way, the proportion of information flowing out is controlled. The expression of the function is described by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (12)$$

The tanh function is derived from the basic hyperbolic functions. It returns a value in the interval from -1 to 1 to control the increase or decrease of the information. Its function expression is described by

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (13)$$

In LSTM, three sigmoid functions manage three gates, namely forget gate, input gate and output gate. LSTM adds the element of memory state C . The current memory state C_t is

determined by the portion of the previous memory state C_{t-1} filtered through the forgetting gate plus the portion added in the current period. The portion that is filtered in the previous period depends on the forgetting gate based on the sigmoid function. When the value of the forgetting gate is 0, it means that the previous memory is completely forgotten. While when the value of the forgetting gate is 1, the previous memory is completely retained. The output formula of the forgetting gate is described by

$$f_t = \sigma(w_f \cdot [H_{t-1}, x_t]) + b_f \quad (14)$$

where H_{t-1} is the model output in period $t - 1$, x_t is the model input in period t , w_f and b_f is the weight and bias of the forgetting gate, respectively, and f_t is the output of the forgetting gate in period t .

The input gate is used to update the new memories. The sigmoid function controls the proportion of updated information and the tanh function controls the magnitude and direction of the update. The two portions are multiplied to get the new memory of the current period. Then, the portion of the previous memory is added to get the memory state of the current period. The formula is described by Equations (15)–(17).

$$i_t = \sigma(w_i \cdot [H_{t-1}, x_t]) + b_i \quad (15)$$

$$u_t = \tanh(w_u \cdot [H_{t-1}, x_t]) + b_u \quad (16)$$

$$l_t = i_t \cdot u_t \quad (17)$$

where w_i and b_i are the weight and bias of the information update ratio, respectively, i_t is the output of the sigmoid function of the input gate in the t th period, w_u and b_u are the weight and bias of the information update magnitude and direction, u_t is the output of the tanh function of the input gate in the t th period and l_t is the output of the input gate in the t th period.

The sigmoid function of the output gate and the tanh function of the current memory are multiplied to obtain the output H_t for the current period. C_t and H_t will flow cyclically to the next period and participate in the calculation. The formula is described by Equations (18)–(20).

$$o_t = \sigma(w_o \cdot [H_{t-1}, x_t]) + b_o \quad (18)$$

$$c_t = c_{t-1} \cdot f_t + l_t \quad (19)$$

$$H_t = o_t \cdot \tanh(c_t) \quad (20)$$

where w_o and b_o are the weight and bias of the output gate, and o_t is the output of the output gate in period t .

2.4. The Architecture of VSNet

The deep learning architecture built in this paper is called VSNet whose architecture is shown by Figure 5. Three different sizes of convolutional kernels are used to extract the features of the samples in parallel. Convolutional kernels of the first size are shaped as a horizontal bar, and extract the sequential features of each vehicle signal in a single sample. As the training progresses, kernels of this size look for a timing variation, which is common to all signals. Convolutional kernels of the second size are shaped as a vertical bar, which tends to extract the relationship features between each vehicle signals related to vehicle speed in a single sample. Convolutional kernels of the third size are a rectangle, which extracts the combined features of signals and sequences. After the process, a LSTM layer is used to extract the sequential features between the sample. Eventually, final results of three types convolution is concatenated by the concatenation layer. The output is performed by a two-layer fully connected neural network to predict the vehicle speed.

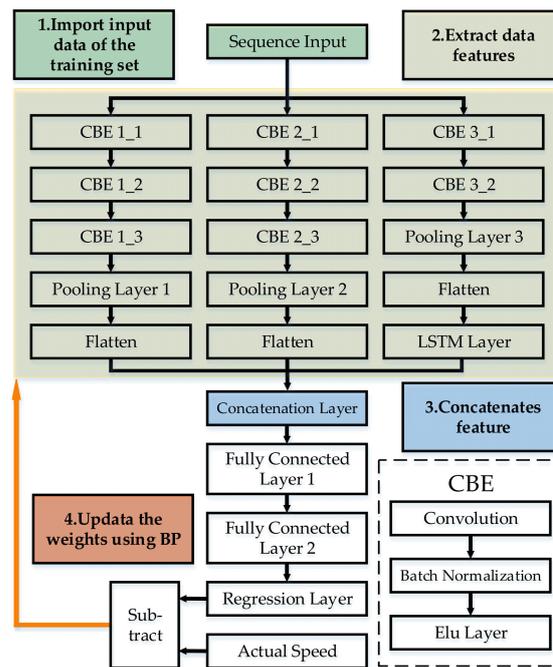


Figure 5. The architecture of VSNet.

The input data format of VSNet is shown by the expression of Equations (2) and (3) above. The input data with time series are combined to constitute the training set, as shown by Equation (21).

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} \quad Y = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{Bmatrix} \quad (21)$$

where X and Y are the input and output of the training set, respectively.

In the initial stage of model training, the initial weights W_0 of the model is set randomly. After inputting the training set, the model brings X and W into the calculation for speed prediction, as shown by

$$\hat{Y} = VSNet(X, W) = \begin{Bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_n \end{Bmatrix} \quad (22)$$

where \hat{Y} is the set of prediction results. The prediction results are compared with the actual output results Y to verify the model prediction accuracy, as shown by

$$S = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (23)$$

where S is the sum of the squared errors. The goal of training is to minimize S by continuously adjusting W . The W is adjusted using the back propagation method and the adjustment magnitude of each weight is calculated by gradient descent method. The calculation formula is shown by

$$\Delta w_j = -\eta \frac{\partial S}{\partial w_j} \quad (24)$$

where Δw_j is the step size of once adjustment of the weight and η is the learning coefficient. In order to find the minimal value of S , it is necessary to correct W gradually from its initial value to the value where the partial derivative of S is 0. The chain rule is used to solve the partial derivative of the complex function S . The error is propagated from the output layer by layer and no intermediate steps can be skipped in this process.

W_0 tends to produce a large error. The calculation based on its derivative also results in a large initial step size. As the error function is adjusted so that S gradually converges to a minimal value, the absolute value of the derivative gradually converges to 0 and the step size is continuously reduced. If η is set too small, the learning speed is too slow and takes too long. On the contrary, if η is set too large, it is very likely to skip the extreme point and continue to oscillate without convergence. Therefore, η is adopted 1.5×10^{-3} in this paper to improve the learning speed while ensuring convergence to the extremum.

The update of weights in VSNet is completed in this way.

3. Model

In this paper, a HEV is modeled in Simulink platform and a MPC-based energy management strategy is built.

3.1. Vehicle Model

The vehicle model is a parallel HEV. According to the placement of the electrification component, the vehicle belongs to the P2 type, i.e., the motor is placed behind the clutch 0 and before the transmission. The vehicle configuration is shown by Figure 6. When the clutch 0 is disengaged, the motor drives the vehicle alone, avoiding engine operating in the inefficiency area and allowing for brake energy recovery. When the clutch 0 is engaged, the engine and motor work together. According to the power distribution relationship between the engine and the motor, the vehicle driving model can be divided into motor traction model, engine traction mode, charging mode and hybrid traction mode. The switching conditions between modes are mainly determined by the overall vehicle demand power and the state of charge (SOC). Vehicle and component parameters are shown by Table 2.

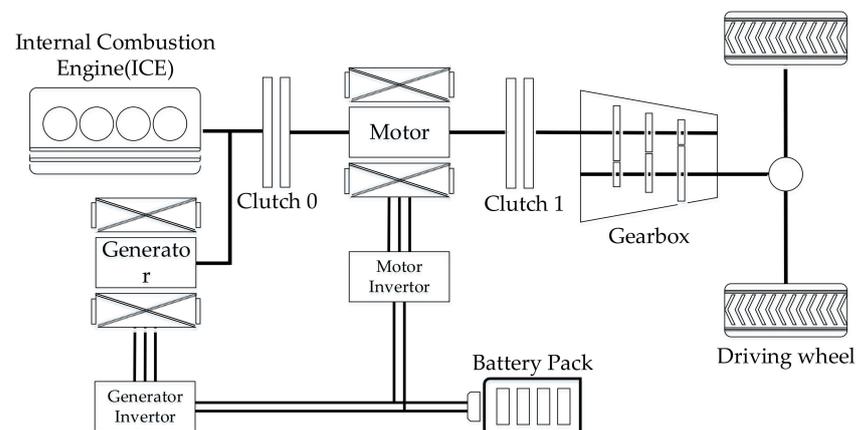


Figure 6. The schematic of the PHEV configuration.

Table 2. Vehicle and component parameters in the studied vehicle.

Component	Parameter	Value	Unit
Vehicle	Vehicle weight	1200	kg
	Tire radius	0.323	m
	Final drive ratio	4.021	/
Engine	Maximum torque	165	Nm
	Maximum power	105@6500	kW@rpm
Motor	Maximum torque	307	Nm
	Maximum power	126@12,584	kW@rpm
Battery	Rated capacity	20.8	Ah
	Rated voltage	366	V
Gear	First	3.527	/
	Second	2.025	/
	Third	1.382	/
	Fourth	1.058	/
	Fifth	0.958	/

3.2. Model Predictive Control

MPC, as a control method that allows rolling optimization, is widely used in industrial control processes [41]. The mechanism of MPC is described below. At each sampling moment, a finite stages open-loop optimization problem is solved online based on the obtained current measurement information. The first element of the obtained control sequence is applied to the controlled object. At the next stage, the optimization problem is refreshed and solved again by using the new measurements as the initial conditions for predicting the future dynamics of the system.

3.2.1. State Update

Predicting the future state of the system, as shown by Figure 7. The system variables, given control variable and parameters are input at each stage. The system variables and outputs for the next stage can be calculated by combining the input with the vehicle model. The system variables, control variable and parameters are defined as shown by Equation (25). The differential equation of the system is described in Equation (26).

$$\begin{cases} x = [v, a, Q_m, Q_e]^T \\ u = \lambda \\ p = [v^p, gear]^T \end{cases} \quad (25)$$

$$\begin{cases} \dot{x} = f(x, u, p, t) \\ y = g(x, u, p, t) \end{cases} \quad (26)$$

where x is the system variable, v is vehicle speed, a is acceleration, Q_m is motor energy consumption, Q_e is engine energy consumption, u is the control variable, λ is power distribution ratio, p is parameter, v^p is target speed and $Gear$ is the gear of the transmission. The vehicle equation constraint for the objective function is shown by

$$\begin{cases} \frac{dv}{dt} = a \\ \frac{da}{dt} = \frac{a_k - a_{k-1}}{step} \\ \frac{dQ_m}{dt} = P_m \\ \frac{dQ_e}{dt} = P_e \end{cases} \quad (27)$$

where a_k is the acceleration of the k th stage, $step$ is the step size of the integration, P_m is the motor power and P_e is the engine power.

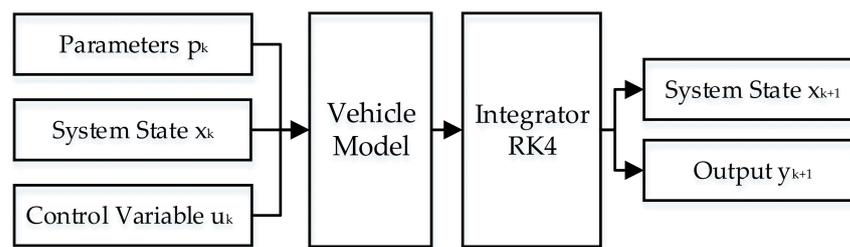


Figure 7. Illustration of the status update process.

In order to solve the differential equations, the fourth-order Longacurta method is used in this paper, which has high accuracy, converges faster and does not require the calculation of higher order derivatives [42]. The calculation formula is shown by

$$\begin{cases} x_{k+1} = x_k + \frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{3} + \frac{K_4}{6} \\ K_1 = hf(x_k, t_k) \\ K_2 = hf(x_k + \frac{K_1}{2}, t_k + \frac{h}{2}) \\ K_3 = hf(x_k + \frac{K_2}{2}, t_k + \frac{h}{2}) \\ K_4 = hf(x_k + K_3, t_k + h) \end{cases} \quad (28)$$

where k is the current stage, f denotes the vehicle model, t is the time, and h is the integration step.

3.2.2. Optimization

Solving the open-loop optimization problem, as shown by Figure 8. The control variable and parameters in the given time domain can continuously update the state variables of the system to obtain the output sequence in the time domain. The output sequence is accumulated to calculate the optimization objective function. The automatic derivative (AD) tool CasADi is used to find the partial derivatives of the objective function with respect to the control variable [43]. The control variable is solved optimally by combining the inequality constraints of the control variable and vehicle components. The optimization method used is a nonlinear primal-dual interior point method, which solves for the optimal sequence of control variable at each moment of the predicted time domain [44].

$$\min f = \min \sum_{i=1}^n y_i = \min \sum_{i=1}^n [\alpha(v_i - v_i^p)^2 + \beta Q_e + \chi Q_m] \quad (29)$$

where f is the objective function, n is horizon; y_i is the output of the i th stage, v_i is the vehicle speed of the i th stage, v_i^p is the target vehicle speed of the i th stage, α , β and χ are the scale factor of each item, and Q_e and Q_m are energy consumption of engine and motor, respectively.

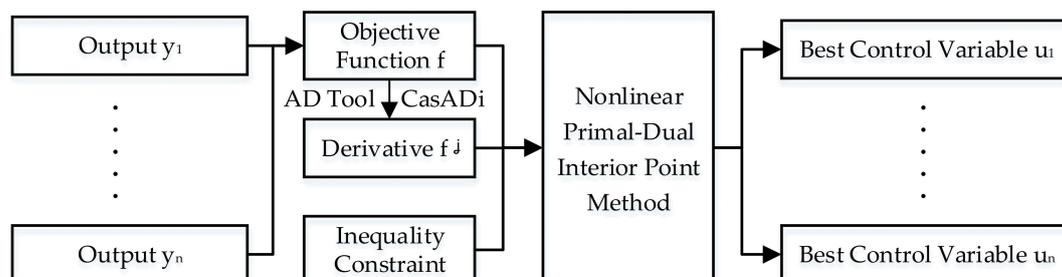


Figure 8. Illustration of the optimization process.

3.2.3. Constraints

The inequality constraints are set up in order to keep the vehicle and its components in a reasonable state and ensure that the results are in the feasible domain. The inequality constraints for the objective function are shown by

$$\left\{ \begin{array}{l} -1 \leq \lambda \leq 1 \\ -1 \leq pedal \leq 1 \\ -T_{m,min}^n \leq T_m^n \leq T_{m,max}^n \\ -T_{e,min}^n \leq T_e^n \leq T_{e,max}^n \\ -n_{m,min} \leq n_m \leq n_{m,max} \\ -n_{e,min} \leq n_e \leq n_{e,max} \end{array} \right. \quad (30)$$

where *pedal* indicates the load of pedal, T_m^n and T_e^n are the torque of motor and engine at the rotate speed of n , respectively, n_m and n_e are the rotate speed of motor and engine, respectively, $T_{m,max}^n$, $T_{e,max}^n$, $n_{m,max}$ and $n_{e,max}$ are the maximum value of the parameters, respectively, $T_{m,min}^n$, $T_{e,min}^n$, $n_{m,min}$ and $n_{e,min}$ are the minimum value of the parameters, respectively.

After determining the constraints and objective function, the solution can be solved by the optimizer. The first element of the optimization solution is applied to the system to complete the calculation of MPC at the current moment and obtain the real vehicle state at the next moment. Finally, the optimization problem is refreshed and the control strategy for the next moment is solved again based on the new information.

4. Results and Validation

Speed prediction can obtain the future vehicle driving state in advance, which is helpful for energy management strategy to make more reasonable control and further develop the vehicle energy saving potential. The speed prediction and simulation experiment adopts a 1300 s driving cycle without training of any model.

4.1. Performance of VSNet

Figure 9 shows the tail figures of the four methods for the test driving cycle. Figure 10 shows the distribution range of the predicted tails for the different methods. Figures 11–14 show the heat maps of the predicted vehicle speed for MCMC, SVM, CNN and VSNet, respectively. Figures 15 and 16 compare the vehicle speed distributions for the first and fifth seconds for the four methods in the form of heat maps. Figure 17 shows the box plots of the speed prediction errors. Figure 18 shows the performance of the four methods.

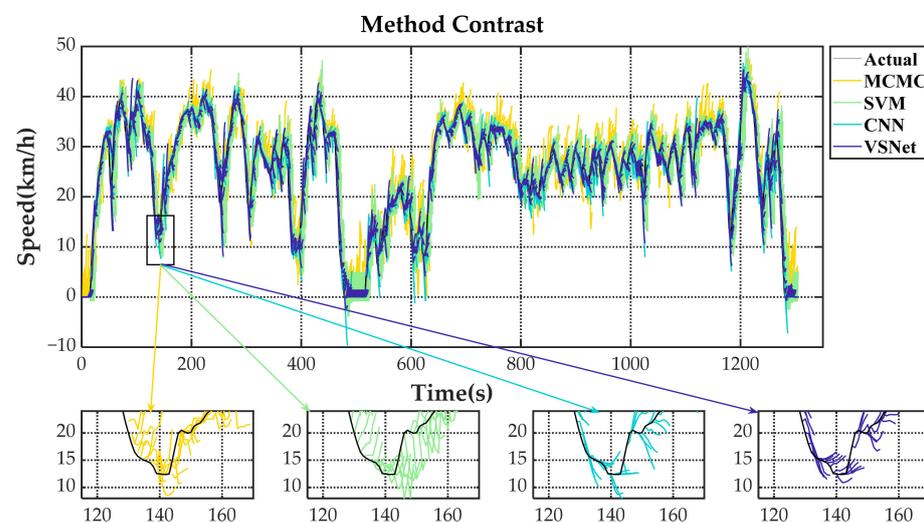


Figure 9. The tail figures of the four methods for the test driving cycles.

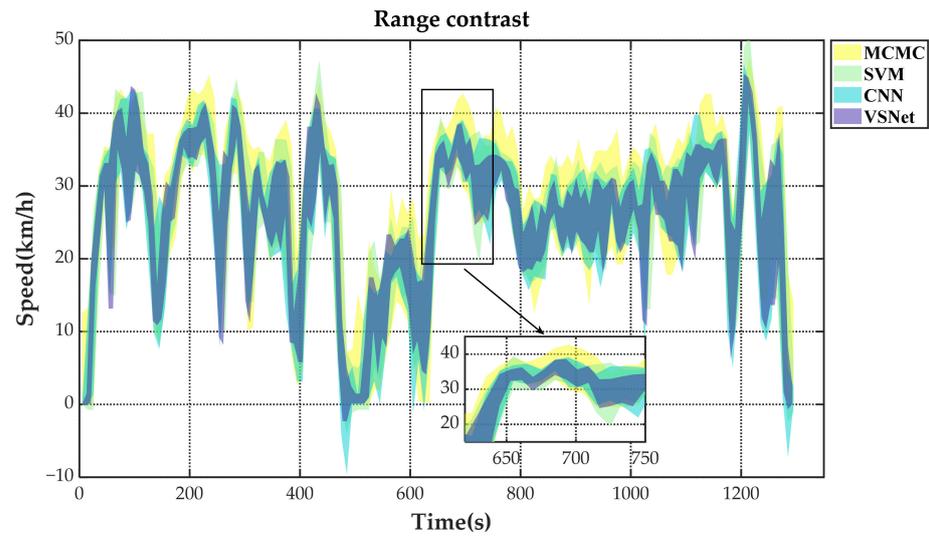


Figure 10. The distribution ranges of the predicted tails of the four methods.

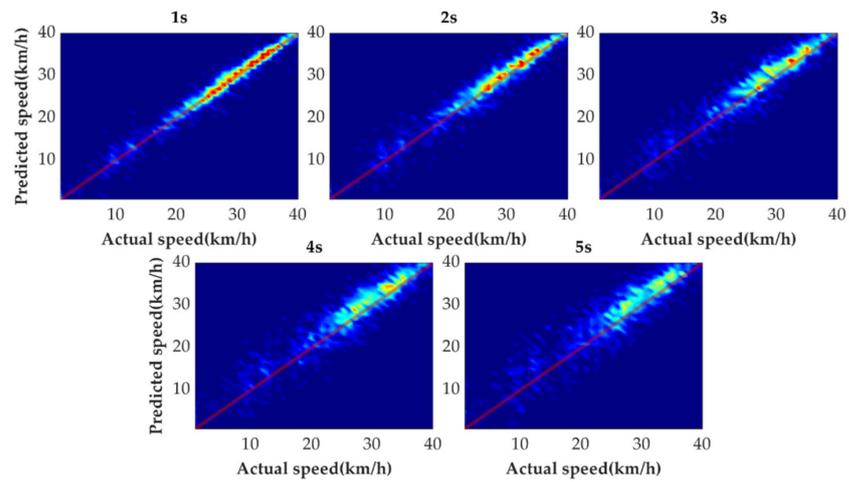


Figure 11. Heat map display of the predicted results of MCMC.

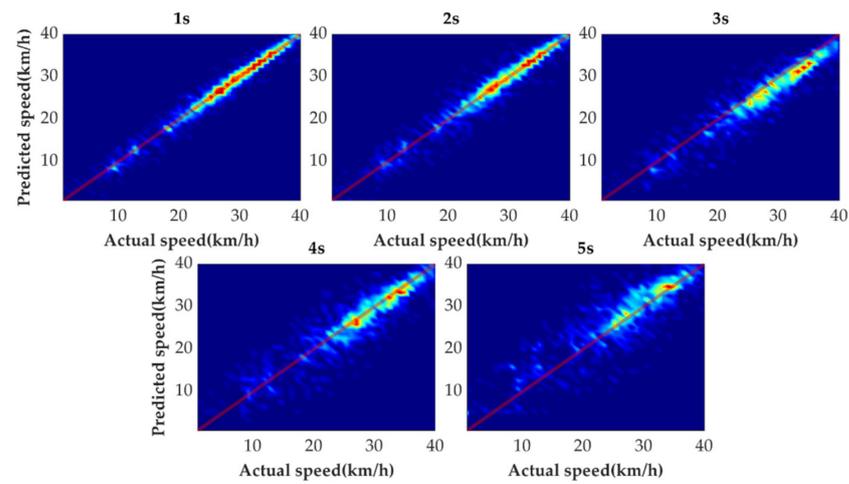


Figure 12. Heat map display of the predicted results of SVM.

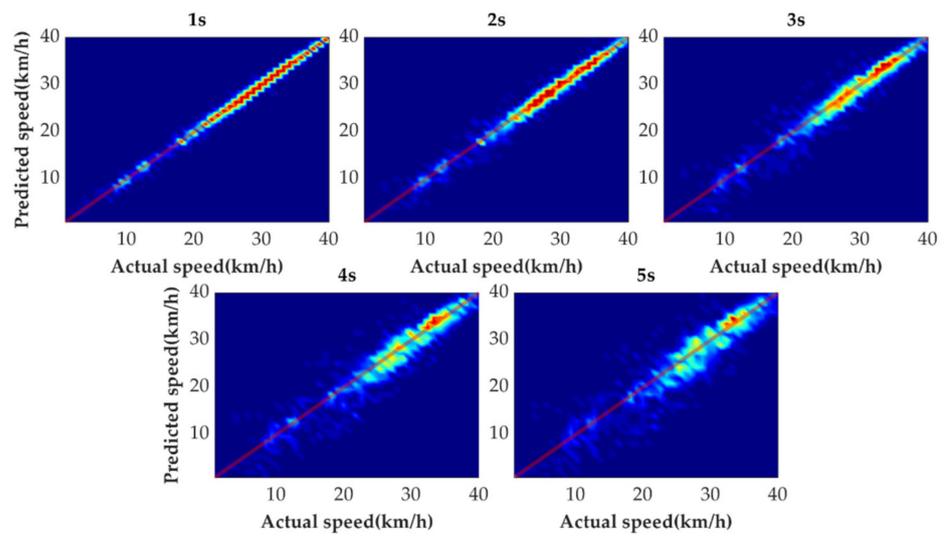


Figure 13. Heat map display of the predicted results of CNN.

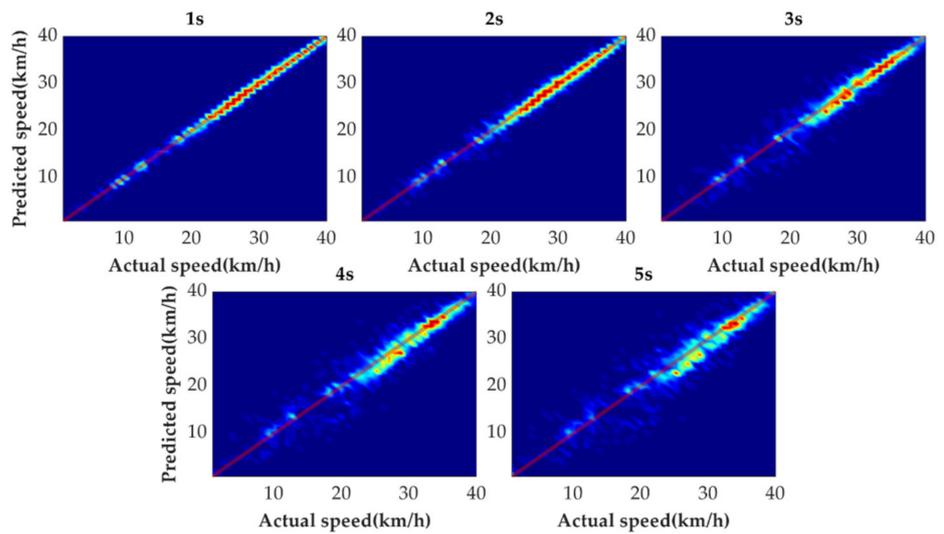


Figure 14. Heat map display of the predicted results of VSNet.

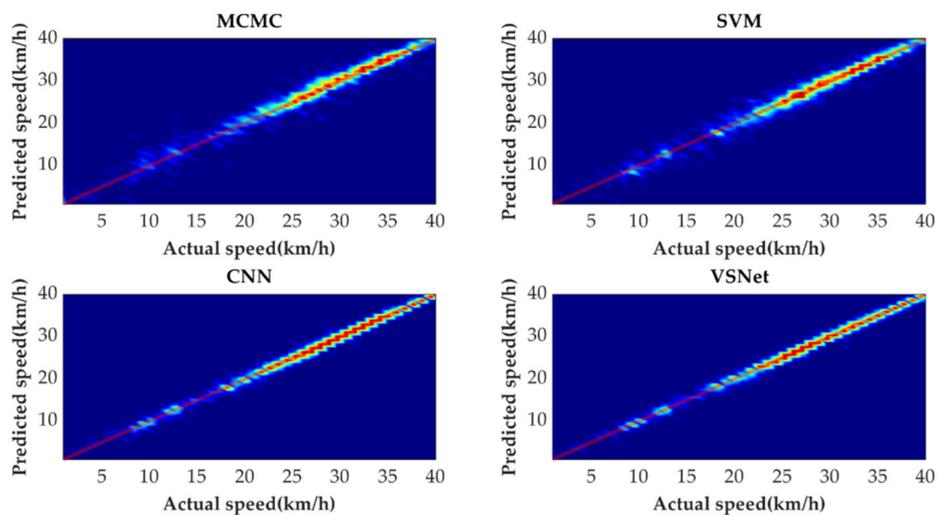


Figure 15. Heat map display of the predicted results for the next 1 s.

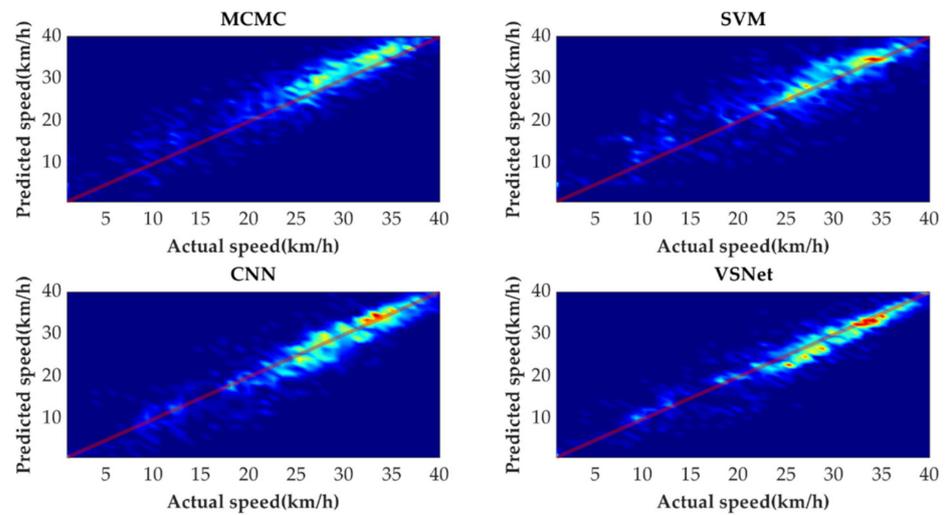


Figure 16. Heat map display of the predicted results for the next 5 s.

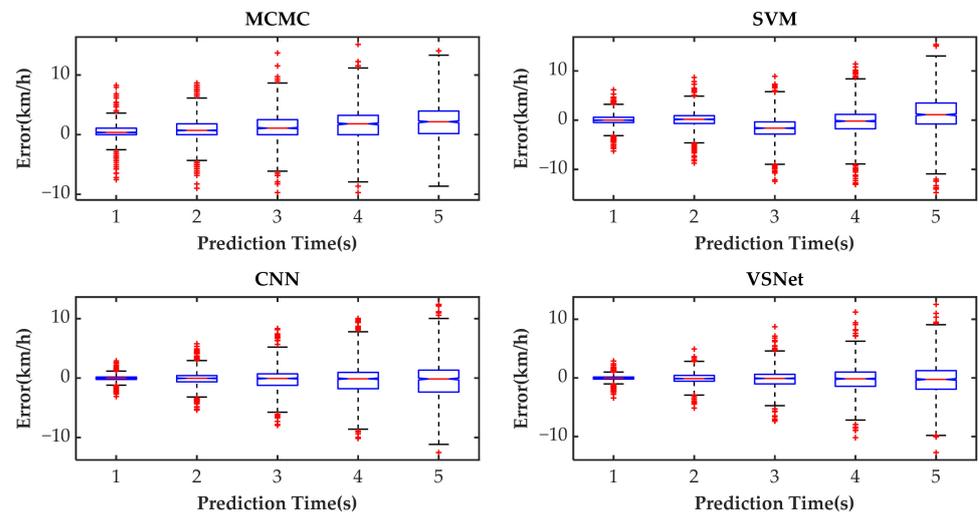


Figure 17. The box plots of the speed prediction errors.

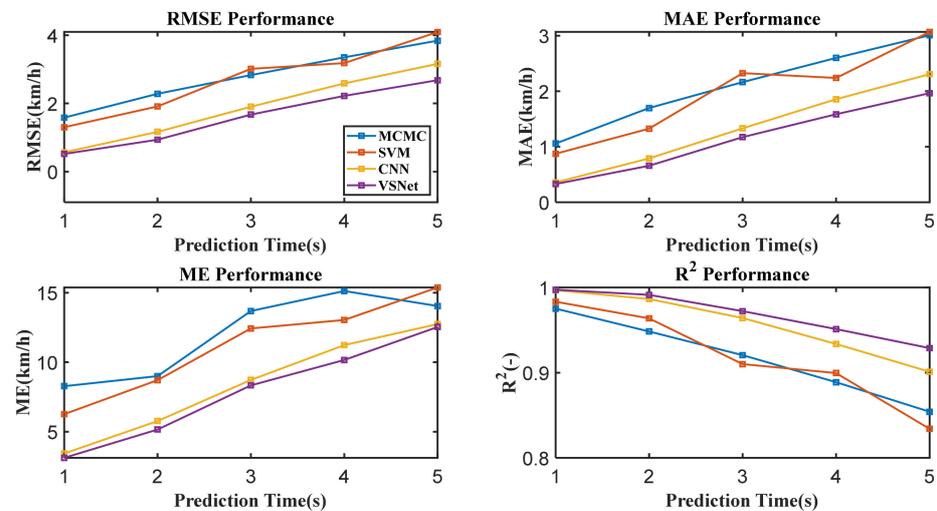


Figure 18. The performance of the four methods.

The black line in Figure 9 is the actual vehicle speed for the test driving cycle. Each colored line is the prediction result of vehicle speed for the next 5 s corresponding to its

starting moment. When the colored curves are closer to the black curve, it means the better prediction result. Figure 9 shows that yellow has the widest coverage, followed by green. The blue and dark purple, on the other hand, have a more concentrated distribution of curves, which can only sometimes be distinguished at the turn of the curve. This also means that MCMC has the worst prediction effect, followed by SVM, while CNN and VSNet have better prediction effect. As can be seen from the partial enlargement of Figure 9, MCMC has a strong randomness in the predicted speed change due to the randomness of the process of moving from one state to another in the state space. The process has the non-aftereffect property, which means future evolution does not depend on past evolution under known current conditions. The detail view of SVM has more regular prediction results compared with MCMC because SVM belongs to supervised learning. Supervised learning minimizes the empirical risk and confidence range by finding the structured risk minimum. In this way, the generalization ability of learning machine can be improved and good statistical rules can be obtained. However, the predicted results during acceleration often show a braking state and the predicted results during deceleration are driven. Such opposite prediction affects the performance of energy management strategy. After adopting the deep learning method, the features of input data can be automatically extracted for prediction by combining the vehicle component signal with the vehicle chassis signal and the prediction accuracy is significantly improved. CNN can quickly identify the current vehicle driving state and give the prediction speed of the corresponding state, which leads to a significant reduction of the blue line coverage. However, the prediction results of CNN have a large error when the vehicle state changes. The prediction results of VSNet have higher accuracy compared with those of CNN. The detail view of VSNet shows that the future state of the vehicle can be predicted before the vehicle state changes, which indicates that VSNet can find a more exact mapping relationship between input and output.

Figure 10 shows the distribution of the ranges of predicted tails. The plotting scheme with transparency has been adopted in order to show the overlapping parts more clearly. The intersection of different regions can be identified by color changes. The original colors of the ranges are shown in the legend. From Figure 10, we can get same conclusions as before. MCMC has the widest coverage and SVM is next, which indicates that the predictions are subject to a wide margin of error. The obvious blue area can be found in the process of vehicle state switching, which indicates that the prediction effect of CNN method is still slightly insufficient, while VSNet achieves the best result. A clear blue area can be found in the process of vehicle state switching, indicating that the prediction effect of CNN is still slightly insufficient. VSNet, on the other hand, achieves the best results among all methods.

In order to further evaluate the prediction results reasonably, the errors of each prediction horizon and various prediction methods are compared next. Figures 11–16 show the heat maps of the different methods and horizons. The colors in the figure represent density. The closer the color is to red indicates the more consistent prediction results at the actual vehicle speed. The closer the distribution is to the boundary of 45 degrees in the first quadrant, the higher the prediction accuracy is. Firstly, the prediction errors of different time horizons are compared. From Figures 11–14, it can be seen that distributions of all prediction methods are gradually deviating from the boundary of 45 degrees as the prediction horizon increases. This represents a gradual increase in prediction error and a significant decrease in prediction accuracy. With the increase of prediction horizon, the driving state of vehicles has more possibilities and the prediction difficulty increases. Moreover, prediction results at the same speed gradually disperse, which leads to a decrease in concentration and even disappearance of the red area. Comparing the high-speed prediction with the low-speed prediction, the results of high-speed prediction of different methods and horizons are better than the results of low-speed prediction. This result arises from the fact that low-speed conditions are mostly caused by environmental constraints, which lead to high complexity and difficulty for prediction. Comparing the different methods, the results of MCMC are slightly higher than the boundary of 45 degrees, especially

for predicting the next 3 to 5 s. This indicates that MCMC is more inclined to predict the accelerated results when the state is transferred. The prediction results of SVM at the 3rd second are significantly smaller compared with the actual speed. From the concentration and distribution, it can be seen that the prediction results of CNN and VSNet based on deep learning methods are significantly better than those of SVM and MCMC. This conclusion is easier to draw in Figures 15 and 16. In addition, it can be seen from Figure 17 that VSNet has extremely high accuracy in predicting the speed in the next 1 s and all the data are almost concentrated around the boundary of 45 degrees.

Figure 17 shows the errors distribution of different prediction methods in the form of the box plot. The middle line of the box, which is the median of the data, represents the average level of the predicted errors. The upper and lower limits of the box, respectively, are the upper and lower quartiles of the predicted errors. Therefore, the width of the box reflects to some extent the degree of fluctuation of the data. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol. All bounds except the median are gradually moved away from 0 km/h as the predicted horizon increases. As the range between the lower and upper boundaries is wider, the outliers beyond the range also gradually decrease. Since there are more acceleration states than braking states during the historical state transfer of the vehicle, the median MCMC gradually increases as the prediction horizon increases. The median of SVM fluctuates around 0 km/h with the third second being the most obvious. The medians of CNN and VSNet are approximately maintained at 0 km/h. The 25th and 75th percentiles of VSNet are obviously closer to 0 km/h, which proves that VSNet has better prediction effect.

Figure 18 shows the RMSE, Mean Absolute Error (MAE), Maximum Absolute Error (ME) and R Squared (R^2) for the different methods at different horizons. It can be observed intuitively from the figure that VSNet outperforms the other three methods in terms of MAE, RMSE, ME and R^2 in all horizons.

4.2. Simulation

In this paper, the vehicle model under the Matlab/Simulink platform is established based on the vehicle configuration in Section 3.1. The vehicle model is equipped with different energy management strategies and simulated for the driving cycle in Section 4.1. The different energy management strategies are the rule-based energy management strategy, the global-optimization-based energy management strategy, and the MPC-based energy management strategies adopting the vehicle speed prediction methods of MCMC, SVM, CNN, VSNet and the known cycle. The known cycle method achieves 100% prediction accuracy by taking the known information of the test driving cycle as the result of speed prediction. The simulation results are analyzed in equivalent fuel consumption, fuel consumption, SOC and operating points of components. The effectiveness of the MPC-based energy management strategy is verified, and the relationship between the speed prediction accuracy and the optimization effect of strategies is explored. The initial SOC set for the simulation is 28% and the vehicle is driven in Charge Depleting (CD) mode during the first half of the simulation. As the power is consumed the vehicle enters Charge Sustaining (CS) mode. The entire process is based on the vehicle speed and the overall vehicle demand power to choose single power source traction mode, hybrid traction mode or charging mode, maintaining the SOC near 25%. The single traction mode, hybrid traction mode or charging mode is selected according to the overall vehicle demand power and SOC to maintain SOC around 25%.

Figure 19 and Table 3 show the equivalent fuel consumption for the simulations with different energy management strategies. Figure 20 shows the fuel consumption during the simulations with different energy management strategies. Figure 21 shows the SOC during the simulations with different energy management strategies. Figure 22 shows the distribution of engine operating points. Figure 23 shows the distribution of motor operating points.

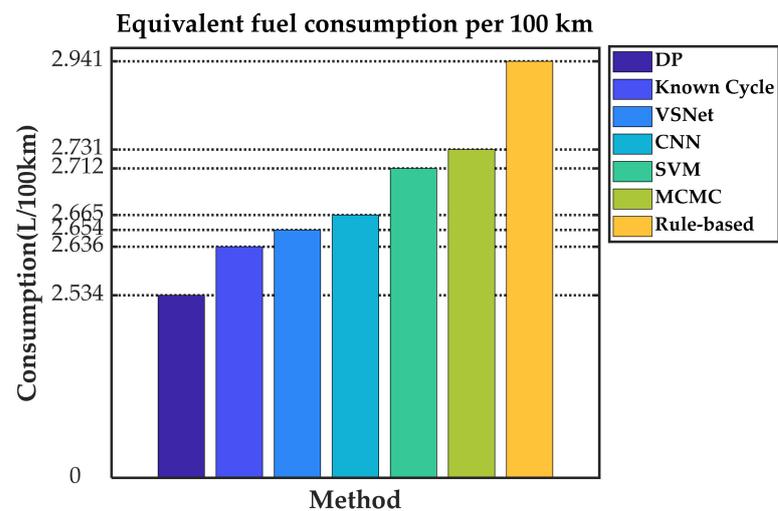


Figure 19. The equivalent fuel consumption for the simulations with different energy management strategies.

Table 3. Simulation results.

Method	Fuel Consumption (mL)	Final SOC (%)	Equivalent Fuel Consumption (L/100 km)	Increased Equivalent Fuel Consumption Compared with DP (%)
DP	141.64	25.00	2.534	/
Known Cycle	145.05	24.79	2.636	4.03
VSNet	149.55	24.88	2.654	4.74
CNN	148.93	24.83	2.665	5.17
SVM	169.17	25.33	2.712	7.02
MCMC	167.03	25.20	2.731	7.77
Rule-based	193.35	25.32	2.941	16.06

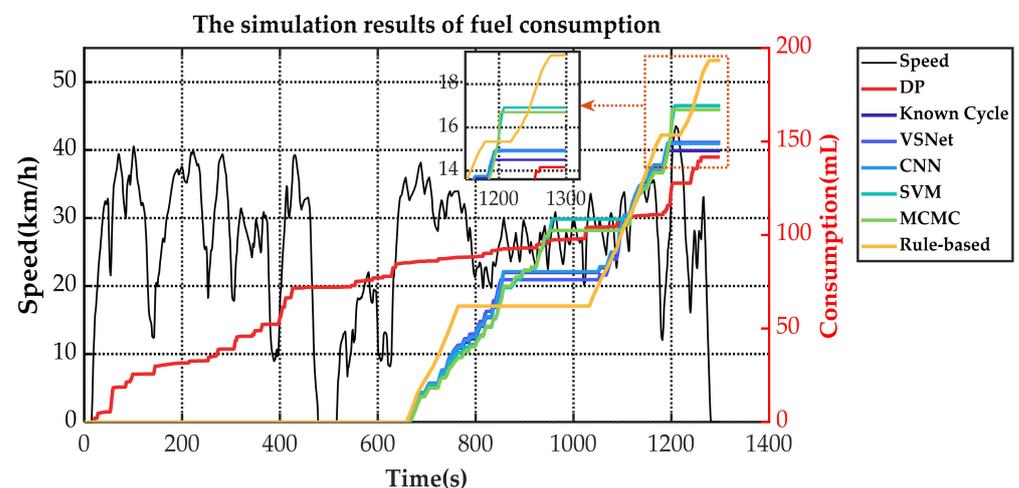


Figure 20. The fuel consumption during the simulations with different energy management strategies.

As can be seen in Figure 19, optimization-based energy management strategies have lower equivalent fuel consumption compared with the rule-based energy management strategy, regardless of the optimization method adopted. As a criterion of optimization effectiveness, DP has the best economy with an equivalent fuel consumption of only 2.534 L/100 km. In the terms of instantaneous-optimization-based energy management strategies, known-cycle-based MPC can be optimized according to real future driving conditions and achieves an equivalent fuel consumption of 2.636 L/100 km with 4.03% more than DP. MCMC-based MPC achieves the highest equivalent fuel consumption in the MPC methods with 7.77% more than DP and 7.14% less than rule-based because of

its fluctuating predicted speed. VSNet, which has the best speed prediction capability, achieved the best MPC-based optimization results except for known cycle. The equivalent fuel consumption of VSNet-based MPC is 2.654 L/100 km, which is 4.74% less than DP and only 0.018 L/100 km higher than known-cycle-based MPC.

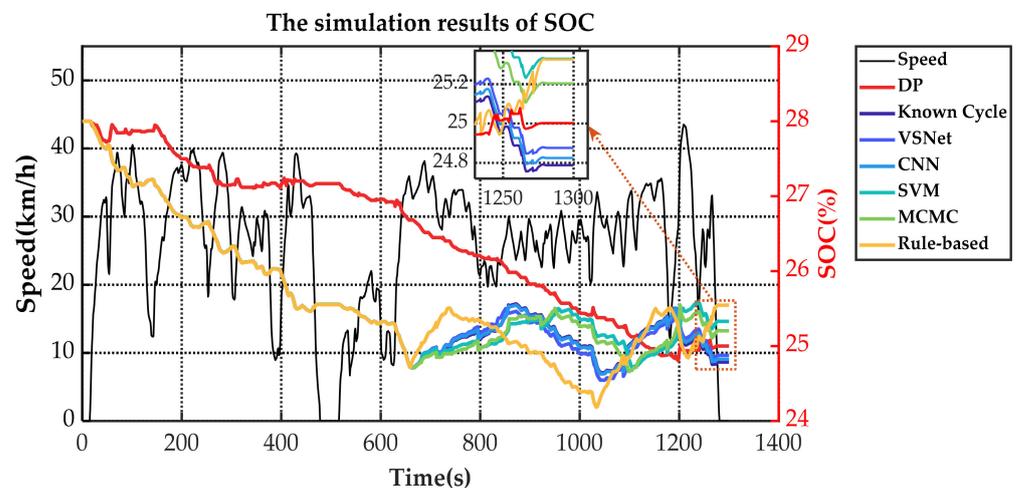


Figure 21. The SOC during the simulations with different energy management strategies.

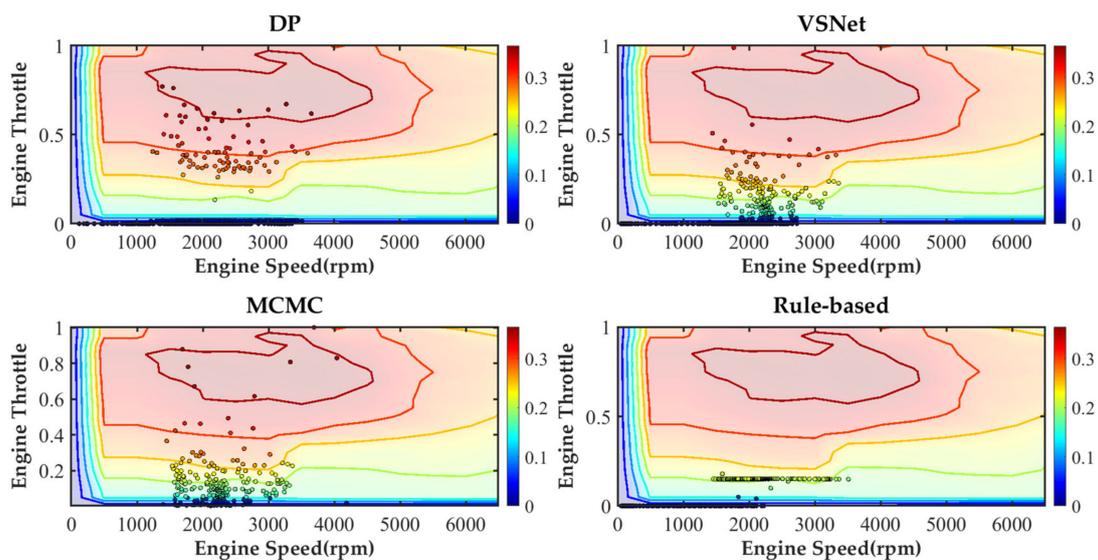


Figure 22. The distribution of engine operating points under various strategies.

The changes of fuel consumption in Figure 20 show that the DP starts the engine frequently to keep the fuel consumption slowly rising. CNN-based MPC, VSNet-based MPC and known-cycle-based MPC approximately maintain the same operating period and fuel consumption. The rule-based energy management strategy obtains the highest energy consumption of all methods. Since not optimized for operating points of the engine, it has an approximately constant fuel consumption rate. Combined with Figure 21, results show that DP often starts the engine to assist the motor in sharing the vehicle demand power. Therefore, its SOC changes are also relatively flat and nearly linearly decreasing. For MPC, the SOC fluctuation range is roughly the same for all five methods. The difference is that the fuel consumption and SOC of MCMC-based MPC and SVM-based MPC increase more gently and charge more slowly. CNN-based MPC and VSNet-based MPC are closer to known-cycle-based MPC. The rule-based energy management strategy has the maximum charging power. In the terminated state, CNN-based MPC, VSNet-based

MPC and known-cycle-based MPC maintain similar SOC. Rule-based energy management strategy, MCMC-based MPC and SVM-based MPC have higher SOC and less electricity consumption at the cost of greater fuel consumption.

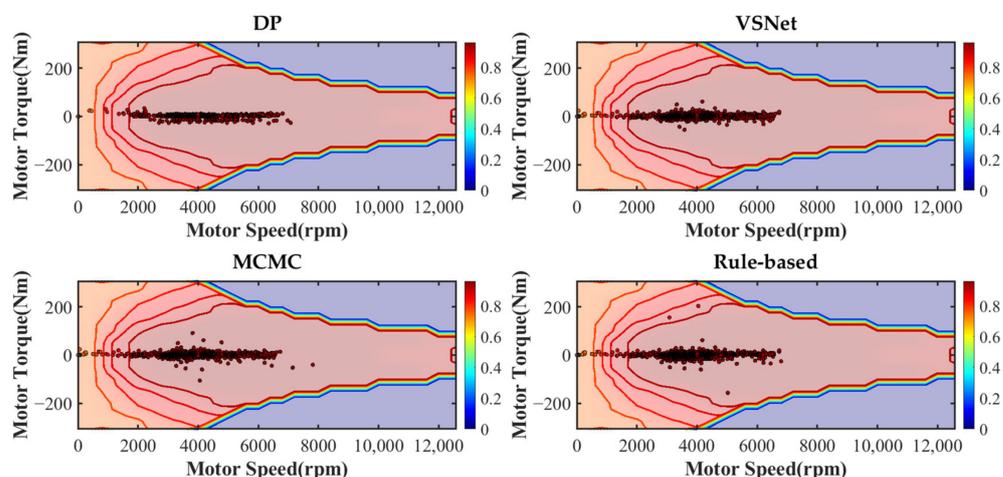


Figure 23. The distribution of motor operating points under various strategies.

In order to further analyze the working condition of components, distributions of engine operating points and motor operating points under DP, VSNet-based MPC, MCMC-based MPC and rule-based energy management strategy are compared. From Figure 22, points represent the engine operating points and colors represent the efficiency of the engine. In the results of DP, the engine adopts a high power to drive the motor to charge the battery each time it is turned on. The large throttle percentage increases the engine loading rate and improves the engine efficiency. MPC does not select engine operating points of high power in order to obtain the smallest objective function in the predicted horizon. The operating points of the motor are only partially distributed in the high efficiency area. VSNet-based MPC relies on high accuracy prediction of vehicle speed to obtain a better distribution of engine operating points compared with MCMC-based MPC. The rule-based energy management strategy has poor regulation capability and more concentrated distribution.

From Figure 23, points represent the motor operating points and colors represent the efficiency of the motor. Observe the distribution, as most of operating points are concentrated in the high-efficiency region, with DP being the most concentrated and VSNet being second. The least effective is the rule-based energy management strategy. Due to an improper power distribution ratio, the motor sometimes undertakes too much power. Some motor operating points have been distributed to the area of low efficient.

5. Conclusion and Future Outlook

In this paper, an architecture for the deep learning-based neural network called VSNet is constructed. VSNet is able to predict future vehicle speed just by self-historical data. A fake image consisting of 15 vehicle signals for the past 15 s is input into VSNet to predict the vehicle speed for the next 5 s. Representative methods from stochastic prediction, machine learning and deep learning are selected for comparison with VSNet. We employ all methods for prediction for a driving cycle that is not involved in any model training. From the prediction results, it can be concluded that RMSE, MAE, ME and R^2 of VSNet are better than the other methods. An MPC energy management strategy based on the speed prediction method is also constructed for simulation and analysis. The simulation results can be summarized that the power optimization effect of MPC is positively correlated with the speed prediction accuracy. With the increase of prediction accuracy of vehicle speed, the difference of MPC compared with DP can be reduced from 7.77% to 4.03%, and the

simulation results are closer to the results of DP in terms of fuel consumption, electrical consumption and power distribution.

In this paper, we design a vehicle speed prediction method with high accuracy for power distribution of multi-power system. In the future, we will build an energy management strategy for complex driving condition based on fused multi-sensor information.

Author Contributions: Conceptualization, J.X. and L.C.; methodology, J.X. and Z.H.; software, J.X. and W.S.; validation, J.X. and L.C.; writing—original draft preparation, J.X.; writing—review and editing, Y.Z. and Z.H.; supervision, L.C. and Z.H.; funding acquisition, W.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Planning Project in Changzhou (CZ20210033).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Gan, L.; Lv, W.; Chen, Y. Capital structure adjustment speed over the business cycle. *Financ. Res. Lett.* **2021**, *39*, 101574. [\[CrossRef\]](#)
- Sokolov-Mladenovic, S.; Milovancevic, M.; Mladenovic, I.; Alizamir, M. Economic growth forecasting by artificial neural network with extreme learning machine based on trade, import and export parameters. *Comput. Hum. Behav.* **2016**, *65*, 43–45. [\[CrossRef\]](#)
- De Moura, J.; Yang, J.; Butt, S.D. Physics-Based Rate of the Penetration Prediction Model for Fixed Cutter Drill Bits. *J. Energy Resour. Technol.* **2021**, *143*, 1–12. [\[CrossRef\]](#)
- Han, S.; Zhang, F.; Xi, J.; Ren, Y.; Xu, S. Short-term Vehicle Speed Prediction Based on Convolutional Bidirectional LSTM Networks. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019.
- Jinquan, G.; Hongwen, H.; Jianwei, L.; Qingwu, L. Real-time Energy Management of Fuel Cell Hybrid Electric Buses: Fuel Cell Engines Friendly Intersection Speed Planning. *Energy* **2021**, *226*, 120440. [\[CrossRef\]](#)
- Ma, D.; Song, X.; Li, P. Daily Traffic Flow Forecasting Through a Contextual Convolutional Recurrent Neural Network Modeling Inter- and Intra-Day Traffic Patterns. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2627–2636. [\[CrossRef\]](#)
- Li, Y.F.; Chen, M.N.; Zhao, W.Z. Investigating long-term vehicle speed prediction based on BP-LSTM algorithms. *IET Intell. Transp. Syst.* **2019**, *13*, 1281–1290. [\[CrossRef\]](#)
- Ibrahim, A.; Goswami, D.; Li, H.; Soroa, I.M.; Basten, T. Multi-layer multi-rate model predictive control for vehicle platooning under IEEE 802.11p. *Transp. Res. Part C Emerg. Technol.* **2021**, *124*, 102905. [\[CrossRef\]](#)
- Wang, X.; Chen, J.; Quan, S.; Wang, Y.-X.; He, H. Hierarchical model predictive control via deep learning vehicle speed predictions for oxygen stoichiometry regulation of fuel cells. *Appl. Energy* **2020**, *276*, 115460. [\[CrossRef\]](#)
- Pei, J.; Su, Y.; Zhang, D.; Qi, Y.; Leng, Z. Velocity forecasts using a combined deep learning model in hybrid electric vehicles with V2V and V2I communication. *Sci. China Technol. Sci.* **2020**, 59–68. [\[CrossRef\]](#)
- Yang, C.; Zha, M.; Wang, W.; Liu, K.; Xiang, C. Efficient energy management strategy for hybrid electric vehicles/plug-in hybrid electric vehicles: Review and recent advances under intelligent transportation system. *IET Intell. Transp. Syst.* **2020**, *14*, 702–711. [\[CrossRef\]](#)
- Li, M.; He, H.; Feng, L.; Chen, Y.; Yan, M. Hierarchical predictive energy management of hybrid electric buses based on driver information—ScienceDirect. *J. Clean. Prod.* **2020**, *269*, 122374. [\[CrossRef\]](#)
- Huang, Y.; Wang, H.; Khajepour, A.; He, H.; Ji, J. Model predictive control power management strategies for HEVs: A review. *J. Power Sources* **2017**, *341*, 91–106. [\[CrossRef\]](#)
- Mirbaha, B.; Saffarzadeh, M.; Beheshty, S.A.; Aniran, M.; Yazdani, M.; Shirini, B. *Predicting Average Vehicle Speed in Two Lane Highways Considering Weather Condition and Traffic Characteristics*; IOP Conference Series: Materials Science and Engineering; IOP Publishing: Bristol, UK, 2017.
- Maji, A.; Sil, G.; Tyagi, A. 85th and 98th Percentile Speed Prediction Models of Car, Light, and Heavy Commercial Vehicles for Four-Lane Divided Rural Highways. *J. Transp. Eng. Part A Syst.* **2018**, *144*, 04018009. [\[CrossRef\]](#)
- Li, L.; Coskun, S.; Zhang, F.; Langari, R.; Xi, J. Energy Management of Hybrid Electric Vehicle Using Vehicle Lateral Dynamic in Velocity Prediction. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3279–3293. [\[CrossRef\]](#)
- Ma, Z.Y.; Zhang, Y.Q.; Yang, J. Velocity and normal tyre force estimation for heavy trucks based on vehicle dynamic simulation considering the road slope angle. *Veh. Syst. Dyn.* **2016**, *54*, 137–167. [\[CrossRef\]](#)
- Shin, J.; Sunwoo, M. Vehicle Speed Prediction Using a Markov Chain with Speed Constraints. *IEEE Trans. Intell. Transp. Syst.* **2018**, *20*, 3201–3211. [\[CrossRef\]](#)

19. Karbowski, D.; Kim, N.; Rousseau, A. Route-Based Online Energy Management of a PHEV and Sensitivity to Trip Prediction. In Proceedings of the 2014 IEEE Vehicle Power and Propulsion Conference (VPPC), Coimbra, Portugal, 27–30 October 2014; pp. 1–6.
20. Li, Y.; Chen, M.; Lu, X.; Zhao, W. Research on optimized GA-SVM vehicle speed prediction model based on driver-vehicle-road-traffic system. *Sci. China* **2018**, *61*, 782–790. [[CrossRef](#)]
21. Hosseini, M.K.; Talebpour, A. Traffic Prediction using Time-Space Diagram: A Convolutional Neural Network Approach. *J. Transp. Res. Board* **2019**, *2673*, 425–435. [[CrossRef](#)]
22. Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transp. Res. Part C Emerg. Technol.* **2015**, *54*, 187–197. [[CrossRef](#)]
23. Yeon, K.; Min, K.; Shin, J.; Sunwoo, M.; Han, M. Ego-Vehicle Speed Prediction Using a Long Short-Term Memory Based Recurrent Neural Network. *Int. J. Automot. Technol.* **2019**, *20*, 713–722. [[CrossRef](#)]
24. Chi, J.T.; Ipsen, I.C. Multiplicative perturbation bounds for multivariate multiple linear regression in Schatten p-norms. *Linear Algebra Appl.* **2021**, *624*, 87–102. [[CrossRef](#)]
25. Hill, G.; Blythe, P.T.; Higgins, C. Deviations in Markov chain modeled electric vehicle charging patterns from real world data. In Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems 2012, Anchorage, AK, USA, 16–19 September 2012; pp. 1072–1077.
26. Hemi, H.; Ghouili, J.; Cheriti, A. Combination of Markov chain and optimal control solved by Pontryagin’s Minimum Principle for a fuel cell/supercapacitor vehicle. *Energy Convers. Manag.* **2015**, *91*, 387–393. [[CrossRef](#)]
27. Sah, S. Machine Learning: A Review of Learning Types. *Preprints* **2020**, 2020070230. [[CrossRef](#)]
28. Cuenca, L.G.; Sanchez-Soriano, J.; Puertas, E.; Andrés, J.F.; Aliane, N. Machine Learning Techniques for Undertaking Roundabouts in Autonomous Driving. *Sensors* **2019**, *19*, 2386. [[CrossRef](#)]
29. Nguyen, H.; Kieu, L.M.; Wen, T.; Cai, C. Deep learning methods in transportation domain: A review. *IET Intell. Transp. Syst.* **2018**, *12*, 998–1004. [[CrossRef](#)]
30. Shrestha, A.; Mahmood, A. Review of Deep Learning Algorithms and Architectures. *IEEE Access* **2019**, *7*, 53040–53065. [[CrossRef](#)]
31. Wang, Z.J.; Turko, R.; Shaikh, O.; Park, H.; Das, N.; Hohman, F.; Kahng, M.; Chau, D.H.P. CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Trans. Vis. Comput. Graph.* **2020**, *27*, 464–474. [[CrossRef](#)]
32. Wang, C.; Hou, Y.; Barth, M. *Data-Driven Multi-Step Demand Prediction for Ride-Hailing Services Using Convolutional Neural Network*; Springer: Cham, Switzerland, 2019.
33. Loaiza, F.A.; Herrera, J.; Mantilla, S.C.L. Using a Separable Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. In Proceedings of the 10th International Conference on Predictive Models in Software Engineering 2018, Sydney, Australia, 8–10 January 2018; pp. 157–161.
34. Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors* **2017**, *17*, 818. [[CrossRef](#)]
35. Qi, X.; Zheng, X.; Chen, Q. *A short term load forecasting of integrated energy system based on CNN-LSTM*. *E3S Web of Conferences*; EDP Science: Les Ulis, France, 2020.
36. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. In *Neural Computation*; MIT Press: Cambridge, MA, USA, 1997; Volume 9, pp. 1735–1780.
37. Pérez-Ortiz, J.A.; Gers, F.A.; Eck, D.; Schmidhuber, J. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Netw.* **2003**, *16*, 241–250. [[CrossRef](#)]
38. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006.
39. Graves, A.; Jaitly, N.; Mohamed, A.-R. Hybrid speech recognition with Deep Bidirectional LSTM. In Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 8–12 December 2013; pp. 273–278.
40. Feng, W.; Zhu, Q.; Zhuang, J.; Yu, S. An expert recommendation algorithm based on Pearson correlation coefficient and FP-growth. *Clust. Comput.* **2018**, *22*, 7401–7412. [[CrossRef](#)]
41. Borhan, H.; Vahidi, A.; Phillips, A.M.; Kuang, M.L.; Kolmanovsky, I.V.; Di Cairano, S. MPC-Based Energy Management of a Power-Split Hybrid Electric Vehicle. *IEEE Trans. Control Syst. Technol.* **2011**, *20*, 593–603. [[CrossRef](#)]
42. Karim, S.A.A.; Ismail, M.T.; Hasan, M.K.; Sulaiman, J. Data interpolation using Runge Kutta method. In Proceedings of the 25th National Symposium on Mathematical Sciences (Sksm25): Mathematical Sciences as the Core of Intellectual Excellence, Pahang, Malaysia, 27–29 August 2017; AIP Publishing: Melville, NY, USA, 2018.
43. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi: A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2019**, *11*, 1–36. [[CrossRef](#)]
44. Yamashita, H.; Yabe, H. Local and superlinear convergence of a primal-dual interior point method for nonlinear semidefinite programming. *Math. Program.* **2010**, *132*, 1–30. [[CrossRef](#)]