IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 March 2025, accepted 20 March 2025, date of publication 26 March 2025, date of current version 2 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3554803

RESEARCH ARTICLE

An RNN-Based Adaptive Hybrid Time Series Forecasting Model for Driving Data Prediction

JI HWAN SEO AND KYOUNG-DAE KIM, (Member, IEEE)
Department of Electrical Engineering and Computer Science, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu 42988, Republic of Korea
Corresponding author: Kyoung-Dae Kim (kkim@dgist.ac.kr)

ABSTRACT In this paper, we propose a hybrid time series forecasting model, named as the Adaptive Multivariate Exponential Smoothing–Recurrent Neural Networks (AMES-RNN), which enables accurate prediction for time series data with non-seasonal and additive trend characteristics. The AMES-RNN follows a hybrid approach in which each of the statistical and deep learning models predicts particular time series components and then merges their output. To enhance prediction performance, the optimal smoothing coefficients of the Exponential Smoothing (ES) model are estimated and updated online. Here, the coefficient estimation is performed through a deep learning-based regression model, and a method for training the regression model is presented. In addition, the prediction model utilizes future-implying information as additional input if available in order to improve prediction accuracy. The effectiveness of the proposed model was validated through multistep forecast tests using vehicle driving data that has non-seasonal and additive trend characteristics. The results show that the prediction accuracy of the proposed model was improved at least 23.0% compared to those of the existing prediction model. Additionally, we demonstrated that AMES-RNN requires low computational resources, making it feasible to perform online predictions.

INDEX TERMS Adaptive, exponential smoothing, hybrid model, RNN, time series forecasting, vehicle data.

I. INTRODUCTION

Time series forecasting is a method of predicting future values based on the patterns of data observed over time [1]. This forecasting approach is model-free, meaning it does not require knowledge of the complex dynamics of the system [2], [3]. Additionally, it can handle multivariate data, making it effective in analyzing correlations among multiple variables [4], and supports multi-step forecasting. Due to these advantages, time series forecasting has been widely used in various domains, such as energy consumption [5], stock price [6], traffic flow [7], etc.

Traditionally, statistical models such as Auto-Regression (AR) [8], Auto-Regression Integrated Moving Average (ARIMA) [9], and Exponential Smoothing (ES) [10] have been used for time series forecasting. However, these statistical models have limitations in capturing complex data patterns [11]. Accordingly, interest in applying deep learning models that can handle nonlinear relationships within time series data has increased [12]. Among various deep learning models, Recurrent Neural Networks (RNNs) have become a popular choice for time series forecasting due to their ability to remember historical data and capture temporal dependencies [13]. These models usually show superior forecasting performance over other network types [14]. Recently, Transformers have also been applied to time series forecasting as they excel at handling long-term dependencies [15]. However, using only deep learning models can lead to overfitting and over-parameterization problems, even when sufficient training data is available, and this can result in lower prediction accuracy compared to statistical methods [16], [17]. One approach to mitigate these issues for deep learning models is to apply data preprocessing. Instance normalization is a widely used technique that standardizes each input time series data by subtracting its mean and dividing by its standard deviation [18]. Through this process, the input data has consistent distributions in

The associate editor coordinating the review of this manuscript and approving it for publication was Sajid Ali.

terms of mean and variance, helping the deep learning models to focus more on predicting temporal patterns. As demonstrated in [18] and [19], this technique has been utilized in various studies. Another approach is employing hybrid method, which use both statistical and deep learning models to complement the limitations of each method. Hybrid approaches have also shown improved prediction accuracy in various fields [20], [21], [22].

This study aims to improve the prediction accuracy for time series data in the vehicle system domain where time series forecasting techniques have been utilized in various ways such as enhancing fuel consumption efficiency [23], [35], advancing driving safety [24], [29], [30], [31], improving user experience in driving simulators [32], and extending vehicle life cycles through early detection of abnormal status [34]. To achieve this goal, we adopt a hybrid approach instead of instance normalization, which can have some limitations when applied to driving data. Specifically, in cases where the driving data remains nearly constant (e.g., constant speed driving), the standard deviation is close to 0. In this case, applying instance normalization can transform the data as if it has extreme variability, providing distorted information to the model. Moreover, this normalization process, which makes the mean and variance across all input data the same, loses information about the strength of trend: Two data with similar shapes of patterns but different rates of change (e.g., smooth cornering vs. sharp cornering) will appear almost identical after instance normalization. This may hinder the model's ability to utilize diverse driving behaviors in forecasting. For these reasons, we leverage the hybrid method for predicting driving data. Additionally, we employ RNN models as deep learning models for the following two reasons. First, driving data is difficult to collect on a large scale due to cost and variability in driving conditions, especially when developing individual driver-oriented models. In such small datasets, RNN-based models are more advantageous than Transformers in achieving high prediction accuracy [25], [26]. Second, RNNs require less memory and computational load compared to Transformers [27], which allows RNNs to be more suitable for vehicle applications where online processing is important but computational resources are limited. On another note, driving data often shows the characteristics of non-seasonal and additive trends. Seasonality is not considered because it is difficult to expect periodic data, which has a clear and regular cycles, unless specific situations are imposed such as slalom tests. Additionally, the change rate of the vehicle states is adjusted by the driver's control inputs, and such adjustments are not necessarily determined depending on the current state values. Hence, additive trend, which assumes the rate of change is independent from current state value, is more preferred than multiplicative trend, where the two components are coupled. Furthermore, since various types of vehicle data that can be collected while driving are inherently related to each other [28], it is very important to capture the interrelations between data types in order to make accurate predictions.

Taking into account all these issues on various types of forecasting models and the characteristics of time series data in the vehicle application domain, we propose a new time series forecasting model, named as the *Adaptive Multivariate Exponential Smoothing - Recurrent Neural Networks (AMES-RNN)* model, which provides better prediction performance than other models for time series data with non-seasonal and additive trend characteristics. The proposed model adjusts its parameters online if the characteristics of incoming time series data change so that the model can provide optimal forecasting performance as possible. Additionally, it leverages informative data that hints at the future to reduce uncertainty in the predictions. The effectiveness of the proposed predictor is validated through multi-step prediction tests on multiple vehicle control signals and the results demonstrate that the prediction accuracy of the proposed predictor is improved compared to those of the existing forecasting techniques.

The remainder of this paper is organized as follows. Section II presents a review of existing time series forecasting methods to improve the prediction accuracy in vehicle applications. Section III introduces MES-LSTM, the foundational model for this research, and discusses its limitations. Section IV describes our approach. Section V presents the experimental results for performance evaluation. Finally, this work is concluded in Section VI.

II. RELATED WORKS

As stated earlier, time series forecasting models have been effectively utilized to enhance the performance of various functions related to vehicles. Guo et al. [23] proposed a method to predict vehicle speed and road gradient by using an ARIMA model. The ARIMA model fitted with pre-acquired driving data is utilized to perform online predictions during vehicle driving. Their tests demonstrated that the maximum Root Mean Square Error (RMSE) for a 10-second-long prediction was about 3.5 m/s for speed and 0.04% for road gradient. However, using fixed time series model parameters limits the ability to reflect online changes in data characteristics during vehicle operation. To address this, Zhang and Fu [29] proposed an online ARIMA model, which continuously updates its parameters to minimize the difference between the data acquired online and the model's predicted values. In a 1.5-second-long prediction test, the online ARIMA model showed maximum RMSE values of 0.27 m for lateral position, 0.40 m for longitudinal position, 0.37 m/s for speed, and 0.40 m/s^2 for acceleration. Although performance comparisons with the traditional ARIMA model were not explicitly provided, the authors showed that the values predicted by online ARIMA could be effectively utilized in a Bi-LSTM-based turning detection system. Additionally, Cao et al. [30] proposed a heuristic method for online updating of the coefficients in an ES model to improve the prediction accuracy of state values for vehicles in sideslip conditions. Through their approach, the prediction accuracy was improved compared to the case

of using fixed coefficients in the 5-second-long prediction test for vehicle acceleration data where rapid trend changes were observed.

In deep learning-based approaches, LSTM networks have been widely used as they can effectively capture long-term dependencies. For instance, Jeong et al. [24] used LSTM to predict surrounding vehicles' trajectories consisting of position, heading, and speed, while Jo et al. [31] applied LSTM to predict lateral acceleration. Alsanwy et al. [32] leveraged LSTM for predicting motion signals of a driving simulator. According to their results, LSTM outperformed other models in multi-step prediction tests. Specifically, [24] showed that LSTM achieved improved trajectory prediction accuracy compared to several conventional vehicle model-based predictors. In particular, compared to the model-based predictor referred to as ''CV/Path'' in [24], the RMSE for longitudinal/lateral position, heading, and speed was reduced by 48.7%, 85.5%, 63.1%, and 54.1%, respectively. In addition, [31] and [32] demonstrated that LSTM showed lower prediction errors than other types of neural networks such as CNN or vanilla RNN. As such, LSTM is effective in predicting time series data acquired from vehicle systems.

For further improvement of the prediction accuracy by complementing the shortcomings of statistical models and deep learning models, hybrid approaches that use both types of models together have gained attention. For example, Zhang [33] proposed a hybrid method of ARIMA and Artificial Neural Network (ANN) which uses ARIMA to capture the linear components of time series data and an ANN to predict the nonlinear components. Here, the nonlinear component corresponds to the residual obtained by subtracting the ARIMA-predicted component from the original data. Zhang [33] showed improved prediction accuracy for sunspot data, lynx population data, and exchange rate data, and this approach was later applied to vehicle data by replacing ANN with other neural networks. Alizadeh et al. [34] proposed a method to predict fuel rate, engine torque, and injection control pressure data using ARIMA-WNN (Wavelet Neural Network). As a result of evaluating the prediction performance for each data by Mean Absolute Error (MAE), ARIMA-WNN improved accuracy by 67.17% and 28.11% on average, compared to ARIMA and WNN alone, respectively. Similarly, Wang et al. [35] utilized ARIMA-LSTM to improve vehicle speed prediction accuracy. The model showed that MAE was reduced by at least 61.6% and 60.7% compared to ARIMA and LSTM, respectively, and it also had superior accuracy compared to other neural network models such as vanilla RNN, CNN, and WNN.

However, the ARIMA model assumes that the differenced time series data satisfies stationarity [36], so the role of the ARIMA model may be limited in cases where stationarity cannot be satisfied even after applying differencing multiple times. A noteworthy model to address this problem is ES-RNN proposed by Smyl [37]. ES-RNN is a hybrid model that performs prediction by sequentially applying an ES model and an RNN model, and both models are suitable for processing non-stationary data. ES-RNN is designed based on the idea that each time series data consists of level l, trend b, seasonality s, and noise ϵ (in some cases, the seasonal component may not be contained). The ES model estimates the level and seasonality for a given time series data, and the RNN model predicts a future trend from the past trend. Smyl [37] verified its performance by winning the M4-competition, a time series prediction accuracy competition, by using LSTM as an RNN model. However, since ES-RNN can only be applied to univariate time series data, it is unable to use interrelations between multiple time series for prediction. To address this limitation, Mathonsi and Zyl proposed Multivariate ES-LSTM (MES-LSTM) [38]. The overall prediction process is similar to that of ES-RNN, but it considers interrelation by passing trend components extracted from multiple time series data to a single LSTM model. In forecasting the number of Covid-19 cases and deaths in South Africa, compared to other 6 existing multivariate time series forecasting models, MES-LSTM reduced RMSE by at least 11.3% and 83.9%, respectively. Another hybrid approach that incorporates the ES model for multivariate time series data is ETSformer [39]. ETSformer is a model where the classical Transformer architecture is redesigned into an encoder-decoder structure that models the time series decomposition process of the Exponential Smoothing. ETSformer is trained to extract and predict the time series components used in the ES model (l, b, and s). Both MES-LSTM and ETSformer do not require the data to be stationary and can consider interrelations among multiple data. However, from the perspective of vehicle applications, MES-LSTM is a more attractive approach, as it consists of computationally efficient models, including ES models (a form of linear recursive formula) and RNNs. This makes the MES-LSTM approach more advantageous for online processing. In addition, unlike ETSformer, where a deep learning model is responsible for extracting time-series components, MES-LSTM decomposes the components according to the mathematical formula of exponential smoothing. This enhances interpretability and can contribute to improving the reliability of vehicle-related functions.

In this paper, we propose a multi-step forecasting model for multivariate time series data with non-seasonal and additive trend characteristics to further improve the prediction accuracy. The proposed model is designed based on the MES-LSTM model. Apart from the MES-LSTM model, our model has additional deep learning-based regression model that is incorporated to estimate optimal ES model coefficients online. In addition, our model is designed to utilize future-implying data, if available, that can be acquired during vehicle driving such as forward road shape [40] or eye-gaze data [41], [42]. As demonstrated in [43], incorporating the future-implying data can reduce the uncertainty and hence can help improve the prediction accuracy. We also improve the input data preprocessing and output data postprocessing

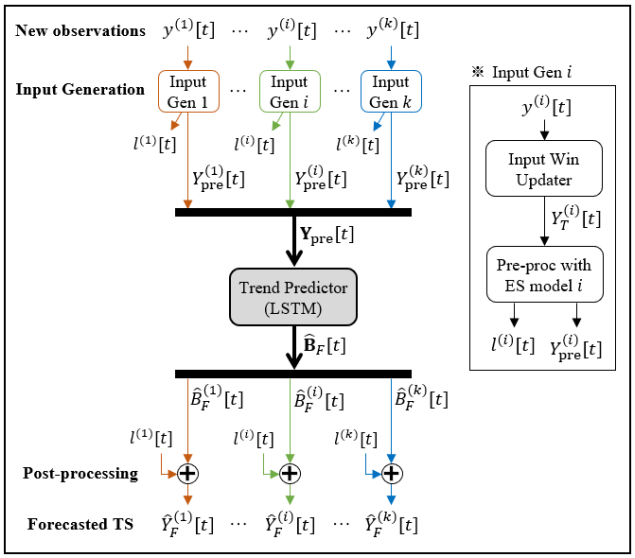


FIGURE 1. A data flow of MES-LSTM for non-seasonal and additive trend time series data.

processes. More details on our model are presented in Section IV.

III. PRELIMINARY

In this section, we briefly review the MES-LSTM model and discuss its limitations especially when it is used with the ES model for non-seasonal and additive trend time series data.

A. MES-LSTM

When configuring MES-LSTM, one ES model is used per each time series data type, and there are various ES models to choose from. The selection of the ES model depends on the trend type (additive or multiplicative trend) and the type of seasonal component (non- or additive or multiplicative seasonality) of the time series data. Among several selectable options for the ES model, we consider the MES-LSTM for non-seasonal and additive trend time series data in this work. The following explanation assumes that non-seasonal and additive trend time series data are processed.

Figure 1 shows the overall data flow of MES-LSTM, and its prediction process is summarized in Algorithm 1. Also, the notations used in Figure 1 and Algorithm 1 are described in Table 1. **ES**$(\cdot)$ in line 6 of Algorithm 1 represents a general exponential smoothing model in (1), which is applicable to non-seasonal and additive trend time series data.

$$\begin{aligned}\hat{y}[t+1|t] &= l[t] + \phi b[t] \\ l[t] &= \alpha y[t] + (1-\alpha)(l[t-1] + \phi b[t-1]) \\ b[t] &= \beta'(l[t] - l[t-1]) + (1-\beta')\phi b[t-1] \\ \text{where} \quad \beta' &= \beta/\alpha \end{aligned} \tag{1}$$

where α and β are the smoothing coefficients for the level and trend components, respectively, and ϕ is the damping factor of the trend component. In general, the smoothing coefficients $\lambda = (\alpha, \beta, \phi)$ satisfy the following inequalities (2), called *admissible constraints*. The conditions (2) are for maintaining the stability of the ES model,

Algorithm 1 MES-LSTM With Non-Seasonal and Additive Trend ES Model

1: **procedure** TS Forecast
2: **init** $\mathbf{Y}_{\text{pre}}[t] \leftarrow$ zeros_like($\mathbf{Y}_T[t]$)
3: **init** $\hat{\mathbf{B}}_F[t] \leftarrow$ zeros_like($\hat{\mathbf{Y}}_F[t]$)
4: **for** $i \in \{1, 2, .., k\}$ **do**
5: $Y_T^{(i)}[t] \leftarrow \left[\, y^{(i)}[t - L_T + 1], \ldots, y^{(i)}[t-1], y^{(i)}[t] \,\right]$
6: $l^{(i)}[t] \leftarrow \mathbf{ES}(Y_T^{(i)}[t],\ \lambda_0^{(i)})$
7: $Y_{\text{pre}}^{(i)}[t] \leftarrow Y_T^{(i)}[t] \ominus l^{(i)}[t]$
8: **end for**
9: $\hat{\mathbf{B}}_F[t] \leftarrow \mathbf{LSTM}(\mathbf{Y}_{\text{pre}}[t])$
10: **for** $i \in \{1, 2, .., k\}$**do** $\hat{Y}_F^{(i)}[t] \leftarrow \hat{B}_F^{(i)}[t] \oplus l^{(i)}[t]$
11: **return** $\hat{\mathbf{Y}}_F[t]$
12: **end procedure**

TABLE 1. Notations in Figure 1 and Algorithm 1.

Symbol	Description
t	Current time step
L_T	Length of input window for time series forecasting
F	Forecast horizon
$f = 1, 2, ..., F$	Forecast step
k	Number of types of time series data
$y^{(i)}[t]$, $l^{(i)}[t]$, $b^{(i)}[t]$	Observed value, level, trend components for each ith type time series $(i \in \{1, 2, .., k\})$
$\lambda_0^{(i)}$	An invariant smoothing coefficient set of the ES model for ith type time series $\lambda_0^{(i)} = (\alpha_0^{(i)}, \beta_0^{(i)}, \phi_0^{(i)})$
$Y_T^{(i)}[t]$	Input window where the observed value of ith type time series is recorded $Y_T^{(i)}[t] = \left[y^{(i)}[t - L_T + 1], ..., y^{(i)}[t]\right]$
$Y_{\text{pre}}^{(i)}[t]$	Preprocessing results for $Y_T^{(i)}[t]$
$\hat{B}_F^{(i)}[t]$	Predicted trend components of ith type data $\hat{B}_F^{(i)}[t] = \left[\hat{b}^{(i)}[t+1|t], ..., \hat{b}^{(i)}[t+F|t]\right]$
$\hat{y}^{(i)}[t+f\|t]$	Predicted value at time step $t+f$ based on data up to time step t
$\hat{Y}_F^{(i)}[t]$	Predicted ith type time series $\hat{Y}_F^{(i)}[t] = \left[\hat{y}^{(i)}[t+1|t], ..., \hat{y}^{(i)}[t+F|t]\right]$
$\mathbf{Y}_T[t]$, $\mathbf{Y}_{\text{pre}}[t]$ $\hat{\mathbf{B}}_F[t]$, $\hat{\mathbf{Y}}_F[t]$	Collection of the corresponding data $\mathbf{Y}_T[t] = \{Y_T^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$ $\mathbf{Y}_{\text{pre}}[t] = \{Y_{\text{pre}}^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$ $\hat{\mathbf{B}}_F[t] = \{\hat{B}_F^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$ $\hat{\mathbf{Y}}_F[t] = \{\hat{Y}_F^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$
$\oplus, \ominus$	Operator: For vector $\mathbf{v} = [v_1, v_2, ..., v_n]$ and scalar a, $\mathbf{v} \oplus a = [v_1 + a, v_2 + a, ..., v_n + a]$ $\mathbf{v} \ominus a = [v_1 - a, v_2 - a, ..., v_n - a]$

ensuring that observations from the distant past have a negligible effect on current and future predictions [44].

$$1 - {}^{1}/_{\phi} < \alpha < 1 + {}^{1}/_{\phi} \tag{2a}$$
$$\alpha(\phi - 1) < \beta < (1 + \phi)(2 - \alpha) \tag{2b}$$
$$0 < \phi \leq 1 \tag{2c}$$

In MES-LSTM, a modified constraint equation (3) is used instead of (2). This defines a region for λ that is a subset of the original region defined by (2).

$$\begin{aligned} &0 < \alpha \leq 1 \\ &0 < \beta < \alpha \quad \text{and} \quad \phi = 1 \end{aligned} \tag{3}$$

Then, within this smaller region for λ, a smoothing coefficient set $\lambda_0 = (\alpha_0, \beta_0, \phi_0) = (\alpha_0, \beta_0, 1)$ is determined and fixed through an optimization process that minimizes the overall difference between $\hat{y}[t+1|t]$ estimated by (1) and the actual $y[t+1]$.

B. LIMITATIONS OF MES-LSTM

First of all, the MES-LSTM model utilizes a fixed smoothing coefficient set λ_0. The model with the fixed coefficient set decomposes the time series components adequately when the time series data to be predicted has similar characteristics to that of the data used to determine λ_0. However, it does not ensure that the time series components are properly estimated when the characteristics of the data are changed over time. Consequently, the LSTM model uses such inaccurately extracted trend components to predict the future, leading to degraded prediction accuracy. This limitation constrains the adaptability of the model in environments where the characteristics of the data are changed dynamically.

Additionally, a constant value of $\phi = 1$ is employed regardless of the characteristics of the time series data. Since this assumption does not reflect the damped trend, it negatively impacts the model's performance when processing data with a damped trend. Adjusting ϕ according to the characteristics of the data would be crucial to improve the prediction accuracy in situations where trend behavior changes.

Appropriate use of the ES model's forecast value is also important for accurate predictions. Equation (4) represents the future time series data that the model aims to predict. As in (4), the sequence $Y_F^{(i)}[t]$ begins with $y^{(i)}[t+1]$.

$$Y_F^{(i)}[t] = \left[y^{(i)}[t+1], y^{(i)}[t+2], \ldots, y^{(i)}[t+F]\right] \tag{4}$$

Subsequently, the prediction for $Y_F^{(i)}[t]$ is computed as described in (5).

$$\begin{aligned} \hat{Y}_F^{(i)}[t] &= \left[\hat{y}^{(i)}[t+1|t], \hat{y}^{(i)}[t+2|t], \ldots, \hat{y}^{(i)}[t+F|t]\right] \\ &= \left[\hat{y}^{(i)}[t+1|t], \hat{y}^{(i)}[t+1|t] + \hat{b}^{(i)}[t+2|t], \right. \\ &\qquad \left. \ldots, \hat{y}^{(i)}[t+1|t] + \hat{b}^{(i)}[t+F|t]\right] \\ &= \left[0, \hat{b}^{(i)}[t+2|t], \ldots, \hat{b}^{(i)}[t+F|t]\right] \oplus \hat{y}^{(i)}[t+1|t] \\ &= \hat{B}_F^{(i)}[t] \oplus \hat{y}^{(i)}[t+1|t] \end{aligned} \tag{5}$$

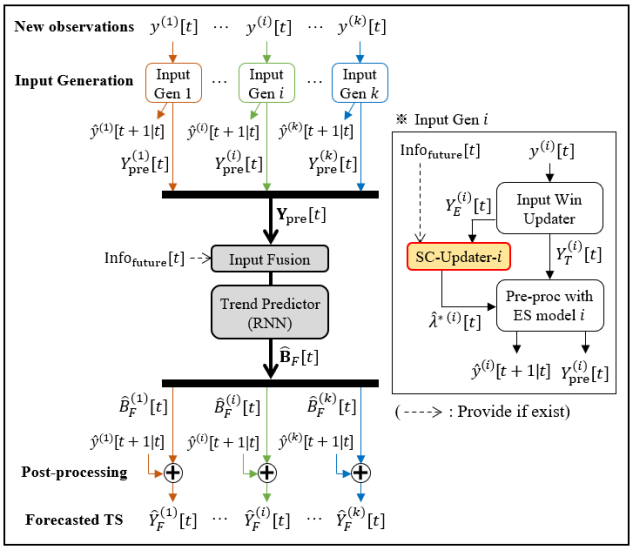


FIGURE 2. A data flow of AMES-RNN for non-seasonal and additive trend time series data.

Referring to (5), for more accurate prediction, it is desirable to use $\hat{y}^{(i)}[t+1|t]$ (the estimated value of $y^{(i)}[t+1]$) as the initial forecasted value for $Y_F^{(i)}[t]$. However, as outlined in line 10 of Algorithm 1, the MES-LSTM model uses $l^{(i)}[t]$ as $y^{(i)}[t+1]$ when computing $\hat{Y}_F^{(i)}[t]$. This approach may lead to potential errors in the prediction process, which can adversely affect the prediction accuracy.

Lastly, MES-LSTM relies only on past-to-current observations to make predictions. This limits its ability to integrate information that may hint at future data. Hence, a way to integrate future-implying data into the prediction process is needed to improve overall prediction performance.

IV. ADAPTIVE MES-RNN

In this section, we present our approaches to address the limitations of MES-LSTM discussed above, aiming to achieve more accurate time series prediction.

Figure 2 shows the overall structure of our proposed time series forecaster, and the prediction process is summarized in Algorithm 2. Most of the notations used in Figure 2 and Algorithm 2 are the same as in Table 1, but changed or newly added notations are described in Table 2. Since the overall prediction process is similar to that of the MES-LSTM, we focus on explaining the main differences from MES-LSTM.

A. SMOOTHING COEFFICIENT UPDATER

First of all, to enhance the adaptability of the model and effectively handle time series data with damped trends, we introduce the Smoothing Coefficient Updater (SC-Updater). When handling a total of k types of time series data, the SC-Updater estimates the optimal smoothing

Algorithm 2 AMES-RNN With Non-Seasonal and Additive Trend ES Model

procedure TS Forecast
  init $\mathbf{Y}_{\text{pre}}[t] \leftarrow \text{zeros\_like}(\mathbf{Y}_T[t])$
  init $\hat{\mathbf{B}}_F[t] \leftarrow \text{zeros\_like}(\hat{\mathbf{Y}}_F[t])$
  for $i \in \{1, 2, .., k\}$ **do**
    $Y_E^{(i)}[t] \leftarrow [\, y^{(i)}[t - L_E + 1], \ldots, y^{(i)}[t-1], y^{(i)}[t] \,]$
    $\hat{\lambda}^{*(i)}[t] \leftarrow$ **SC-Updater-***i*$(Y_E^{(i)}[t], \text{Info}_{\text{future}}[t])$
    $Y_T^{(i)}[t] \leftarrow [\, y^{(i)}[t - L_T + 1], \ldots, y^{(i)}[t-1], y^{(i)}[t] \,]$
    $\hat{y}^{(i)}[t+1|t] \leftarrow \textbf{ES}(Y_T^{(i)}[t],\ \hat{\lambda}^{*(i)}[t])$
    $Y_{\text{pre}}^{(i)}[t] \leftarrow Y_T^{(i)}[t] \ominus \hat{y}^{(i)}[t+1|t]$
  end for
  $\hat{\mathbf{B}}_F[t] \leftarrow \textbf{RNN}(\mathbf{Y}_{\text{pre}}[t], \text{Info}_{\text{future}}[t])$
  for $i \in \{1, 2, .., k\}$**do** $\hat{Y}_F^{(i)}[t] \leftarrow \hat{B}_F^{(i)}[t] \oplus \hat{y}^{(i)}[t+1|t]$
  return $\hat{\mathbf{Y}}_F[t]$
end procedure

TABLE 2. Additional notations in Figure 2 and Algorithm 2.

Symbol	Description
L_E, L_T	Length of input window for time series forecasting L_E: For smoothing coefficients estimation L_T: For time series forecasting
$\text{Info}_{\text{future}}[t]$	Information that implies the near future acquired at time step t
$\lambda^{*(i)}[t]$	Time-varying, optimal smoothing coefficient set of the ES model for ith type time series $\lambda^{*(i)}[t] = (\alpha^{*(i)}[t], \beta^{*(i)}[t], \phi^{*(i)}[t])$
$\hat{\lambda}^{*(i)}[t]$	Estimation of $\lambda^{*(i)}[t]$ $\hat{\lambda}^{*(i)}[t] = (\hat{\alpha}^{*(i)}[t], \hat{\beta}^{*(i)}[t], \hat{\phi}^{*(i)}[t])$
$Y_E^{(i)}[t], Y_T^{(i)}[t]$	Input window where the observed value of ith type time series is recorded $Y_E^{(i)}[t] = [y^{(i)}[t - L_E + 1], ..., y^{(i)}[t]]$ $Y_T^{(i)}[t] = [y^{(i)}[t - L_T + 1], ..., y^{(i)}[t]]$
$\mathbf{Y}_E[t], \mathbf{Y}_T[t]$	Collection of the input windows $\mathbf{Y}_E[t] = \{Y_E^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$ $\mathbf{Y}_T[t] = \{Y_T^{(i)}[t] \mid i \in \{1, 2, .., k\}\}$

coefficient set $\lambda^{*(i)}[t]$ ($i \in \{1, 2, .., k\}$) online for each type of time series data. Basically, $\hat{\lambda}^{*(i)}[t]$ is initially set to the constant coefficient set $\lambda_0^{(i)}$, which provides adequate prediction accuracy overall, but it is adjusted by $d\hat{\lambda}^{*(i)}[t] = (d\hat{\alpha}^{*(i)}[t], d\hat{\beta}^{*(i)}[t], d\hat{\phi}^{*(i)}[t])$ at each time step if there is a different set that offers improved prediction accuracy.

Figure 3 illustrates the overall architecture of the SC-Updater-*i*. Each SC-Updater-*i* is designed as a regression model that utilizes recurrent neural networks to predict $\hat{\lambda}^{*(i)}[t]$ by synthesizing past and present information. In SC-Updater-*i*, each smoothing coefficient is estimated within a finite range, and the ranges are determined as

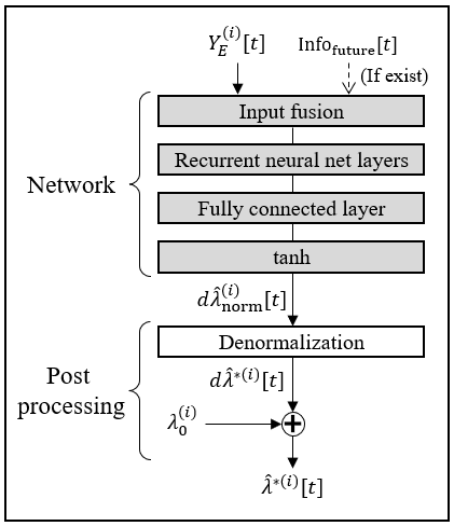


FIGURE 3. A structure of SC-Updater-*i*.

follows. Referring to (2a), the valid range of $\alpha^{(i)}$ expands as $\phi^{(i)}$ decreases. Hence, the range of $\alpha^{(i)}$ that maintains the stability of the ES model regardless of $\phi^{(i)} \in (0, 1]$ is $0 < \alpha^{(i)} < 2$, which is the case when $\phi^{(i)} = 1$. Next, we establish the lower bound for $\beta^{(i)}$. For this, we first rearrange the inequality for $\phi^{(i)}$ in (2c), which results in the following inequality.

$$0 < \phi^{(i)} \leq 1 \quad \Rightarrow \quad -1 < \phi^{(i)} - 1 \leq 0 \tag{6}$$

To ensure model stability, the condition $\alpha^{(i)} > 0$ must be satisfied as above. Then, referring to this and (2b), the lower bound of $\beta^{(i)}$ is obtained.

$$-\alpha^{(i)} < \alpha^{(i)}(\phi^{(i)} - 1) \leq 0 \quad \Rightarrow \quad 0 < \beta^{(i)} \tag{7}$$

Combining all of these, we finally have the following constraint equations for smoothing coefficients that ensures the stability of the ES model.

$$0 < \alpha^{(i)} < 2 \tag{8a}$$
$$0 < \beta^{(i)} \tag{8b}$$
$$\beta^{(i)} < (\phi^{(i)} + 1)(2 - \alpha^{(i)}) \tag{8c}$$
$$0 < \phi^{(i)} \leq 1 \tag{8d}$$

Here, we note that, in comparison to the region for smoothing coefficients defined by (3), the region defined by (8) is much larger while still ensuring the stability of the ES model.

Since each smoothing coefficient is bounded within a specific range, as described in (8), a tanh() layer is incorporated at the end of the network as shown in Figure 3. Through the recurrent neural network, a total of three values $d\hat{\alpha}^{(i)}_{\text{norm}}[t]$, $d\hat{\beta}^{(i)}_{\text{norm}}[t]$, $d\hat{\phi}^{(i)}_{\text{norm}}[t]$ in the range of $[-1, 1]$ are obtained. Then, each $d\hat{\xi}^{(i)}_{\text{norm}}[t]$ ($\xi \in \{\alpha, \beta, \phi\}$) is transformed to exist within the actual range of each smoothing coefficient change $[d\xi^{(i)}_{\text{min}}, d\xi^{(i)}_{\text{max}}]$. Here, the information about minimum and maximum values of each $d\hat{\xi}^{(i)}_{\text{norm}}[t]$ are obtained during the dataset generation process for training SC-Updater. As a

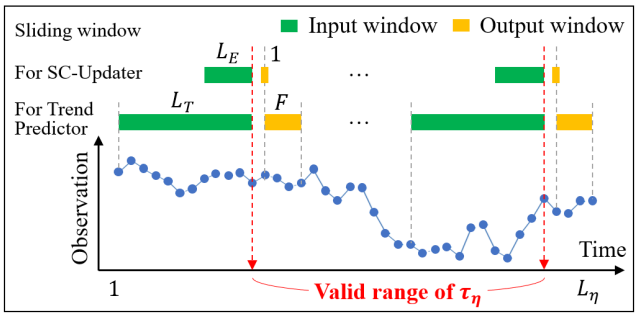


FIGURE 4. Time step range for extracting data samples to train both SC-Updater and Trend Predictor from ηth time series data in the case of $L_T > L_E$ and $F > 1$.

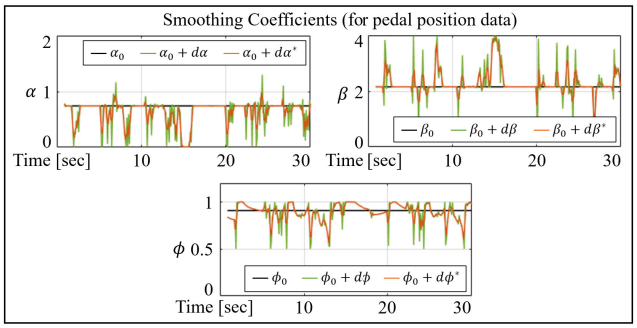


FIGURE 5. An example of smoothing coefficient comparison before and after applying non-causal MA filter.

result, estimated smoothing coefficient changes $d\hat{\lambda}^{*(i)}[t]$ are obtained. Finally, by element-wise adding the default smoothing coefficient set $\lambda_0^{(i)}$ to $d\hat{\lambda}^{*(i)}[t]$, the predicted $\hat{\lambda}^{*(i)}[t]$ is calculated.

B. DATASET GENERATION FOR SC-UPDATER TRAINING

To train the SC-Updater, a dataset is required. For this, we utilize the sliding window technique, which is commonly used when developing deep learning-based prediction models [45]. Let N_{train} be the number of time series data in the training dataset, where each time series data is indexed by $\eta \in \{1, 2, .., N_{\text{train}}\}$. Subsequently, let L_η be the length of ηth time series data, and τ_η be the time index within the ηth time series data ($\tau_\eta = 1, 2, .., L_\eta$). To extract data samples for SC-Updater training from a given ηth time series data, the input window and output window sizes are set to L_E and 1, respectively. Then, the amount of coefficient change at time step τ_η, denoted by $d\lambda^{*(i)}[\tau_\eta]$, is determined by solving the optimization problem in (9).

$$\begin{aligned} \min_{d\lambda^{*(i)}[\tau_\eta]} \ & \left|\hat{y}^{(i)}[\tau_\eta+1|\tau_\eta] - y^{(i)}[\tau_\eta+1]\right| \\ \text{s.t. } & \lambda^{*(i)}[\tau_\eta] = \lambda_0^{(i)} + d\lambda^{*(i)}[\tau_\eta] \\ & \lambda^{*(i)}[\tau_\eta] \text{ satisfies (8)} \\ & \hat{y}^{(i)}[\tau_\eta+1|\tau_\eta] = \mathbf{ES}(Y_E^{(i)}[\tau_\eta],\ \lambda^{*(i)}[\tau_\eta]) \end{aligned} \tag{9}$$

where $\tau_\eta \in \left[\max(L_E, L_T), L_\eta - F\right]$. Note that this range for τ_η is defined as such in order to generate a dataset that is valid for training both the SC-Updater and the Trend Predictor. An illustration for the valid range of τ_η is shown in Figure 4.

Through this optimization problem, the $d\lambda^{*(i)}[\tau_\eta]$ that most accurately predicts the first future value for each input window is found. Once $d\lambda^{*(i)}[\tau_\eta]$ is obtained for all input windows generated from ηth time series data, a non-causal moving average (MA) filter is applied to smooth out the changes in $d\lambda^{*(i)}[\tau_\eta]$, thereby more clearly capturing the changing pattern of the smoothing coefficients as shown in Figure 5. This MA-filtered $d\lambda^{*(i)}[\tau_\eta]$ is used as a label for the input window $Y_E^{(i)}[\tau_\eta]$.

C. LOSS FUNCTION FOR SC-UPDATER TRAINING

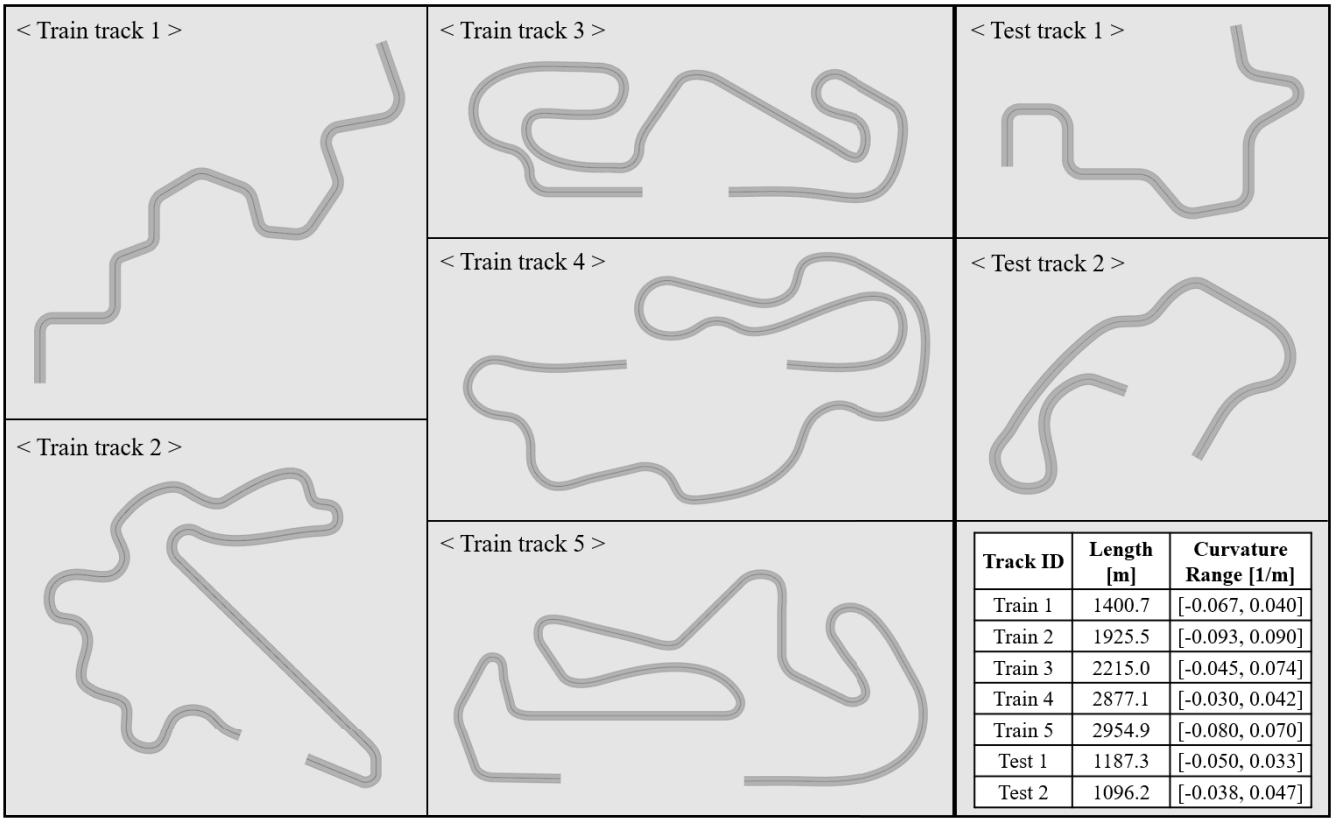
We aim to train the SC-Updater to estimate the smoothing coefficients as accurately as possible while maintaining the stability of the ES model. To this end, the loss function is designed using the difference between $d\lambda^{*(i)}[\tau_\eta]$ and $d\hat{\lambda}^{*(i)}[\tau_\eta]$ and (8c), where $d\hat{\lambda}^{*(i)}[\tau_\eta]$ is the value predicted by SC-Updater-i for $d\lambda^{*(i)}[\tau_\eta]$. To enable the model to accurately estimate the smoothing coefficient set, one of several error measurement metrics, such as Mean Absolute Error (MAE), Mean Square Error (MSE), or Root Mean Square Error (RMSE), can be employed to design the loss function. Each metric offers its own advantages as a loss function: MAE is relatively robust to outliers, MSE aids efficient convergence to minima by leveraging gradually decreasing gradients, and RMSE penalizes errors less severely than MSE [46]. As an example, if MSE is employed, the loss function for the ηth time series is designed as (10).

$$\text{Loss}_{1,\tau_\eta}^{(i)} = \sum_{\xi\in\{\alpha,\beta,\phi\}} (d\xi^{*(i)}[\tau_\eta] - d\hat{\xi}^{*(i)}[\tau_\eta])^2 + \text{StabLoss}_{\tau_\eta}^{(i)} \tag{10}$$

The second term $\text{StabLoss}_{\tau_\eta}^{(i)}$ is for maintaining the stability of the ES model, and it becomes 0 if $\hat{\lambda}^{*(i)}[\tau_\eta]$ satisfies (8c). Otherwise, the value of $\text{StabLoss}_{\tau_\eta}^{(i)}$ increases as the difference between the left- and right-hand sides of (8c) increases. Note that the other inequalities in (8) are not utilized for $\text{StabLoss}_{\tau_\eta}^{(i)}$ design, as these conditions are sufficiently satisfied using the tanh() layer in the SC-Updater-i.

D. PREPARATION OF TREND PREDICTOR TRAINING

The Trend Predictor also requires a dataset and a loss function for training. To this end, a sliding window strategy is employed similarly to the dataset preparation process of the SC-Updater. By setting the input and output window sizes to L_T and F, the input and output windows at time step τ_η, denoted by $Y_T^{(i)}[\tau_\eta]$ and $Y_F^{(i)}[\tau_\eta]$ respectively, are extracted from the ηth time series data in the training dataset. Then, the pre-processed windows $Y_{\text{pre}}^{(i)}[\tau_\eta]$ and $B_F^{(i)}[\tau_\eta]$ are calculated



Track ID	Length [m]	Curvature Range [1/m]
Train 1	1400.7	[-0.067, 0.040]
Train 2	1925.5	[-0.093, 0.090]
Train 3	2215.0	[-0.045, 0.074]
Train 4	2877.1	[-0.030, 0.042]
Train 5	2954.9	[-0.080, 0.070]
Test 1	1187.3	[-0.050, 0.033]
Test 2	1096.2	[-0.038, 0.047]

FIGURE 6. Tracks for driving time series data collection.

by (11).

$$Y^{(i)}_{\text{pre}}[\tau_\eta] = Y^{(i)}_T[\tau_\eta] \ominus \hat{y}^{(i)}[\tau_\eta + 1|\tau_\eta]$$
$$B^{(i)}_F[\tau_\eta] = Y^{(i)}_F[\tau_\eta] \ominus \hat{y}^{(i)}[\tau_\eta + 1|\tau_\eta] \quad (11)$$

where $\tau_\eta \in \left[\max(L_E, L_T), L_\eta - F\right]$, and $\hat{y}^{(i)}[\tau_\eta + 1|\tau_\eta]$ is the first future value estimated by the ES model configured with the *already trained* SC-Updater-i. It is worth mentioning that since the SC-Updater operates independently of the Trend Predictor, the trained SC-Updater can be utilized during the dataset preparation process for the Trend Predictor.

Then, from $Y^{(i)}_{\text{pre}}[\tau_\eta] = \left[b^{(i)}[\tau_\eta - L_T + 1], .., b^{(i)}[\tau_\eta - 1], b^{(i)}[\tau_\eta]\right]$ for all data type i calculated in (11), the input data for the Trend Predictor $\mathbf{Y}_{\text{in}}[\tau_\eta]$ is constructed as (12).

$$\mathbf{Y}_{\text{in}}[\tau_\eta] = \left[\mathbf{b}_{\tau_\eta - L_T + 1}, .., \mathbf{b}_{\tau_\eta - j}, .., \mathbf{b}_{\tau_\eta}\right] \quad (12)$$

where $\mathbf{b}_{\tau_\eta - j} = \left[b^{(1)}[\tau_\eta - j], .., b^{(i)}[\tau_\eta - j], .., b^{(k)}[\tau_\eta - j]\right]$ for $j \in \{0, 1, .., L_T - 1\}$.

The training of the Trend Predictor aims to estimate the trend component as accurately as possible. For this, the loss function is designed using the difference between $\hat{B}^{(i)}_F[\tau_\eta] = \left[\hat{b}^{(i)}[\tau_\eta + 1|\tau_\eta], .., \hat{b}^{(i)}[\tau_\eta + F|\tau_\eta]\right]$ and $B^{(i)}_F[\tau_\eta] = \left[b^{(i)}[\tau_\eta + 1], .., b^{(i)}[\tau_\eta + F]\right]$ for $i \in \{1, 2, .., k\}$, and an example of such loss function is shown in (13).

$$\text{Loss}_{2,\tau_\eta} = \sum_{i=1}^{k} \sum_{f=1}^{F} (\hat{b}^{(i)}[\tau_\eta + f|\tau_\eta] - b^{(i)}[\tau_\eta + f])^2 \quad (13)$$

MSE is utilized in (13) and, as explained in Section IV-C, other error measurement metrics are also possible to be used.

E. OTHER ENHANCEMENTS

Apart from smoothing coefficient adaptation, several other enhancements are also incorporated to further improve the prediction accuracy. First, to reduce potential errors in the prediction process, $\hat{y}^{(i)}[t + 1|t]$ is used instead of $l^{(i)}[t]$ as indicated in lines 9 and 12 of Algorithm 2, where $\hat{y}^{(i)}[t + 1|t]$ is computed using (1). Second, if there is available data that can hint at the future, denoted as $\text{Info}_{\text{future}}[t]$, it is fused with the input time series data $Y^{(i)}_E[t]$ and $\mathbf{Y}_{\text{pre}}[t]$ (referred to as "Input Fusion" in Figures 2 and 3). This fused data is then transmitted as the input of the RNN models, thereby utilizing the future information when making predictions. As an example of data fusion, if $\text{Info}_{\text{future}}[t]$ is in the form of a sequence of length r, it is concatenated as $\left[Y^{(i)}_E[t], \text{Info}_{\text{future}}[t]\right] \in \mathbb{R}^{L_E + r}$ and $\left[\mathbf{Y}_{\text{in}}[t], \text{Info}_{\text{future}}[t]\right] \in \mathbb{R}^{kL_T + r}$, where $\mathbf{Y}_{\text{in}}[t]$ is generated in the form of (12). Finally, there is no restriction on the selection of RNN models. That is, any RNN model, such as vanilla RNN, GRU, or LSTM, can be employed when configuring both the SC-Updater and the Trend Predictor.

V. EXPERIMENTS

This section presents the prediction accuracy results of the time series forecasters. The prediction performance

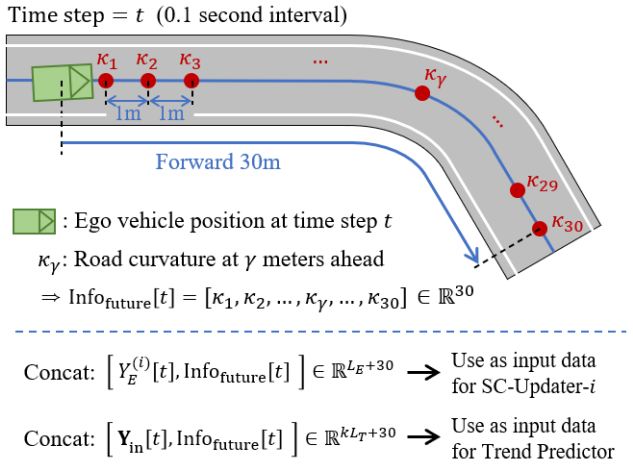


FIGURE 7. An illustration of future-implying data and input data fusion method used in this experiment.

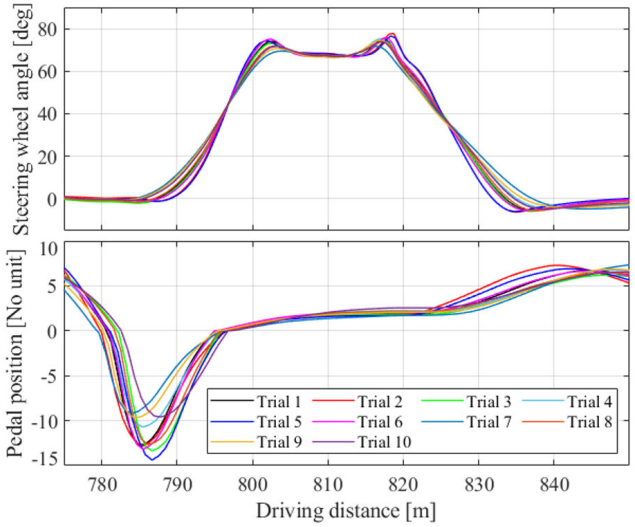


FIGURE 8. An example of acquired driving data for each trial drive run in the same driving direction and road segment[1].

was evaluated on vehicle driving data, which have the non-seasonal and additive trend characteristics.

A. EXPERIMENTAL SETUP

The driving data were collected using IPG CarMaker vehicle simulation software which mimics realistic vehicle movements. IPG CarMaker provides a simulation environment that can configure various driving tracks, and we designed tracks with various curvatures so that diverse forms of time series data are present in the driving data in this experiment. Figure 6 illustrates the overall shapes of the tracks used in the experiments and a summary of information for each track. The dataset for training the SC-Updater and Trend Predictor was obtained using the data collected from driving on the training tracks, and the model performance was evaluated using the data from the test tracks.

The vehicle was controlled by IPG Driver, a virtual driver model included in IPG CarMaker, and was set to drive along the center of the lane on each track. The target data for prediction are the steering angle and pedal position, which are typical input signals used in actual vehicle control. Note that for the pedal position data, the throttle pedal and brake pedal exist as separate control units, but these two signals were combined under the assumption that the driver operates only one pedal at a time. Additionally, for each time step, road curvature information up to 30 meters ahead, recorded at 1m intervals regardless of the current vehicle speed, was provided in the form of sequence as future-implying information. All data were collected at a rate of 10 Hz, and signs (+/−) were assigned to distinguish driving behaviors. For example, '+' indicates left direction for the steering angle and road curvature data and acceleration for the pedal position data. Figure 7 shows the future-implying information used in this simulation and the process of fusing it when generating input data for SC-Updater-i and Trend Predictor.

When collecting the driving data for each track, to obtain diverse data reflecting various factors that can influence driving behavior in real-world environments, we adjusted several parameters that affect the driving behavior of the IPG Driver. Table 3 summarizes the parameters we changed when generating the training and test datasets. For each training track, 10 trial driving runs were conducted in one direction and another 10 in the opposite direction, resulting in a total of 20 trials per track. Similarly, for each test track, 5 trial driving runs were conducted in both directions, totaling 10 trials per track. In every trial conducted on both training and test tracks, all parameter values were randomly set within their specified ranges listed in Table 3, and Figure 8 illustrates the driving data profiles for each trial under the same driving direction and road segment.

Additionally, to mimic sensor noise occurring in an actual environment, we added zero-mean Gaussian noise to the collected driving data. For the steering wheel angle data in this experiment, reflecting the fact that modern vehicles already have a measurement accuracy of 0.1 deg [49], we applied Gaussian noise with a 99.9% probability of noise occurrence within the range of [-0.1 deg, 0.1 deg]. Similarly, for the pedal position and forward path curvature measurements, we employed Gaussian noise with a 99.9% probability of noise occurrence within the range of [-0.5, 0.5] and [-1/25 m^{-1}, 1/25 m^{-1}] respectively, and these ranges were set heuristically.

To find the smoothing coefficient set of the ES model that minimizes the difference between $\hat{y}^{(i)}[t+1|t]$ and $y^{(i)}[t+1]$, we utilized `fmincon()`, which is the optimization tool for nonlinear multivariable functions in MATLAB. However, since `fmincon()` finds a local minimum, it may not find the optimal coefficient set depending on the initial point. To address this issue, when determining the fixed

[1]Note: Pedal position has no units, and its absolute value ranges from 0 and 100, representing fully released and fully pressed, respectively. This description applies to all subsequent contents of the manuscript.

TABLE 3. Simulation parameters that were changed randomly when collecting the driving data.

Parameters	Range	Unit	Description
Road friction coefficient	[0.4, 0.8]	-	This range includes friction coefficients for asphalt and gravel surfaces [47].
Ambient wind speed	[0, 45]	km/h	This range covers conditions from a calm state to a strong breeze [48].
Cut corner coefficient	[0, 0.25]	-	It adjusts the driver to move off the given driving lane while cornering. A higher value tends to increase the deviation and make cornering smoother.
Pedal control smoothing coefficient	[0, 0.25]	-	It adjusts the smoothness of the gas and brake pedal actions. A higher value leads to smoother pedal position adjustment.
Longitudinal accuracy coefficient	[0.9, 1]	-	It determines the accuracy of following the speed profile generated by IPG Driver. A lower value leads to following the speed profile with a higher deviation.
Lateral accuracy coefficient	[0.9, 1]	-	It determines the accuracy of driver steering. A lower value leads to decreased accuracy in maintaining the intended course.

TABLE 4. Network model options.

Parameters	Options
Input window size	10, 15, 20
RNN cell	Vanilla RNN, GRU, LSTM
Hidden state dimension	128, 256, 512, 1024
Loss function to minimize prediction error	Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Square Error (RMSE)

Model Combination	MAE – 99th prctile		Score			Rank
	Pedal [-]	Steer [deg]	Pedal	Steer	Sum	
Comb_1	7.2	6.5	62.8	93.1	155.9	16
Comb_2	5.5	6.9	100	87.5	187.5	**1**
…	…	…	…	…	…	…
Comb_C	6.4	8.1	80.3	70.7	151.0	20

Assign score for each column
(Lowest → 100 / Highest → 0)

FIGURE 9. A method to select the best model configuration using MAE.

coefficient set $\lambda_0^{(i)}$ used in MES-LSTM, we also employed `GlobalSearch()`, a built-in MATLAB tool designed to find the global optimum. The optimal coefficient set $\lambda^{*(i)}[t]$ for each input window required for training SC-Updater was found using only `fmincon()`, with $\lambda_0^{(i)}$ as the initial point. Since the fixed set $\lambda_0^{(i)}$ is determined as the global optimum, we considered that a local search would be sufficient to find $\lambda^{*(i)}[t]$ that corresponds to variations of $\lambda_0^{(i)}$.

The performance of deep learning models is affected by several factors including the network architecture and hyperparameter settings [50]. Therefore, to find the model that provides the most accurate predictions, we applied various model training configurations. Table 4 lists the options used for developing and training the network model in this experiment. For fair comparisons, all models for each combination were trained using Adam optimizer for 15 epochs with a batch size of 10. Both SC-Updater and Trend Predictor were implemented using PyTorch, and the network training was performed on a computer with an Intel i7-11700K CPU, an NVIDIA RTX 3080 Ti GPU, and 64 GB of RAM.

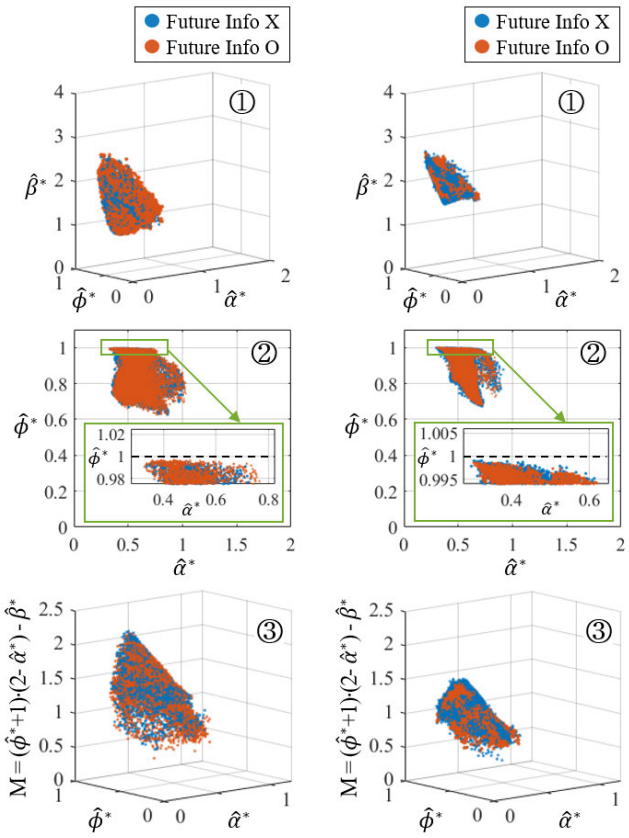
Figure 9 illustrates the process of selecting the Trend Predictor that showed the best prediction results for the test dataset among all possible combinations in Table 4, and the method is as follows. Let C denote the number of combinable model configurations from Table 4. Also, let N_{test} be the number of time series data in the test dataset, where each time series data is indexed by $n \in \{1, 2, .., N_{\text{test}}\}$. Given the nth time series data in the test dataset, first, the multistep predicted time series data generated by the cth model ($c \in \{1, 2, .., C\}$) at each time step, where the forecast horizon is F, were collected. Then, the MAE was calculated from each predicted data using (14), which measures the overall multistep prediction accuracy. The reason for using MAE is that it shows relatively less distortion in the error calculation than other evaluation metrics such as MSE or RMSE.

$$\text{MAE}_c^{(i)}[t] = \frac{1}{F}\sum_{f=1}^{F}\left|\hat{y}_c^{(i)}[t+f|t] - y^{(i)}[t+f]\right| \tag{14}$$

where i denotes a particular data type ($i \in \{1, 2, .., k\}$). Here, the subscript $\cdot_c$ is used to indicate the result obtained from cth model. The MAE values calculated according to (14) at each time step t within the nth time series data were collected for all $n \in \{1, 2, .., N_{\text{test}}\}$ as (15).

$$\mathcal{M}_c^{(i)} = \bigcup_{n=1}^{N_{\text{test}}}\{\text{MAE}_c^{(i)}[t] \mid 20 \le t \le L_n - F\} \tag{15}$$

where L_n is the length of the nth time series data. In order to produce an MAE collection of the same size regardless of the input window size prepared in this experiment, t is set to start from 20, which is the longest input window size. Using this MAE collection, the specific percentile of the MAE was calculated. In this experiment, we used the 99th percentile, representing the approximate maximum error (for convenience, we denote it as $\text{MAE}_{99,c}^{(i)}$ in the following description). Based on the $\text{MAE}_{99,c}^{(i)}$ values, we assigned scores for each data type as indicated by the red boxes in



(a) Pedal position data (b) Steering wheel angle data

FIGURE 10. All smoothing coefficient sets estimated by each SC-Updater for the entire test dataset.

Figure 9. Let $\text{MAE}^{(i)}_{99,\min}$ and $\text{MAE}^{(i)}_{99,\max}$ denote the lowest and highest values among $\{\text{MAE}^{(i)}_{99,c} \mid c = 1, 2, .., C\}$, respectively. We set 100 points to the result with $\text{MAE}^{(i)}_{99,\min}$ (best) and 0 points to the result with $\text{MAE}^{(i)}_{99,\max}$ (worst). The score for cth model was calculated using linear interpolation as (16).

$$\text{Score}^{(i)}_c = 100 \times \frac{\text{MAE}^{(i)}_{99,\max} - \text{MAE}^{(i)}_{99,c}}{\text{MAE}^{(i)}_{99,\max} - \text{MAE}^{(i)}_{99,\min}} \tag{16}$$

Once scores were assigned for all data types, a total score was calculated for each combination. The combination with the highest total score was selected as the best model.

The process of selecting the best SC-Updater was similar to the method described above, but there were two differences. First, since the SC-Updater is designed to accurately predict $y^{(i)}[t+1]$, we set $F = 1$ in (14). Second, the best SC-Updater-i was selected individually for each data type. That is, the process of calculating the total score was omitted when selecting the best SC-Updater-i.

B. VERIFICATION OF ES MODEL STABILITY CONDITION

To verify that the trained SC-Updater estimates smoothing coefficient set that satisfies the stability conditions of the ES model, we checked whether the estimated values of the SC-Updater $\hat{\lambda}^* = (\hat{\alpha}^*, \hat{\beta}^*, \hat{\phi}^*)$ were within the inequalities in (8). Figure 10 visualizes the smoothing coefficient sets estimated by the SC-Updater for each driving data type on the entire test dataset using a scatter plot.

First, the satisfaction of inequalities (8a), (8b), and (8d) can be confirmed through plots ① and ② in each subfigure. Here, plot ② corresponds to the top view of plot ①, and the green box in plot ② is a zoomed-in view around $\phi = 1$ to check if $\hat{\phi}^*$ ever exceeds 1. Plot ① clearly shows that $\hat{\beta}^*$ is always positive, which indicates that (8b) is satisfied. Also, through plot ②, we can confirm that all $\hat{\alpha}^*$ and $\hat{\phi}^*$ are within the range defined by (8a) and (8d).

Next, the satisfaction of inequality (8c) can be verified through plot ③. For reference, the display range of $\hat{\alpha}^*$ in plot ③ is reduced to [0, 1.2], as all $\hat{\alpha}^*$ were within this range in this experiment, as illustrated in plot ②. To easily check whether (8c) was satisfied, we utilized the value $M = (\hat{\phi}^*+1)(2-\hat{\alpha}^*)-\hat{\beta}^*$ shown in plot ③. This value represents the difference between the left- and right-hand sides of (8c), where $M > 0$ indicates that (8c) is met. As shown in the figure, all $\hat{\lambda}^*$ satisfy $M > 0$. To sum up these results, since all inequalities in (8) were satisfied, we confirmed that the proposed SC-Updater can estimate smoothing coefficients that satisfy the stability condition of the ES model.

C. PREDICTION ACCURACY COMPARISON

The AMES-RNN introduces several changes compared to MES-LSTM. These changes include (i) substitution of $l[t]$ with $\hat{y}[t+1|t]$, (ii) expansion of the smoothing coefficients range to (8), (iii) the use of SC-Updater, and (iv) the use of future-implying data. To check the effect of each modification, we measured the prediction error by adding each change one by one to the MES-LSTM. The prepared predictor types for comparison are summarized in Table 5. Here, the predictors referred to as Type 1 and Type 2 in Table 5 are incremental models towards AMES-RNN. Specifically, Type 1 is a modified model of MES-LSTM, replacing $l^{(i)}[t]$ with $\hat{y}^{(i)}[t+1|t]$ in lines 7 and 10 of Algorithm 1. Type 2 is a modified model of Type 1, where the optimal invariant smoothing coefficient set $\lambda^{(i)}_0$ is determined within the range defined by (8) instead of (3), and used in configuring the ES model.

The overall prediction performance of each predictor type was assessed using the 90th and 99th percentiles of the MAE distribution, which represent the typical error level observed in most predictions and the approximate maximum error, respectively. Table 6 presents the prediction performance metrics for each model type in forecasting vehicle control signals. The results indicate that the overall prediction accuracy was improved as each modification was applied, and ultimately, the AMES-RNN+ showed the best performance.

First, we can check the effect of using $\hat{y}[t+1|t]$ by comparing MES-LSTM and Type 1. Referring to the calculation of $\hat{y}[t+1|t]$ in (1), when using $l[t]$, the difference $y_{\text{diff}}[t+1|t] = \hat{y}[t+1|t] - l[t] = \phi b[t] + \text{noise}[t]$ was

TABLE 5. Descriptions of each time series forecaster type and corresponding best model option.

Predictor	Use $l[t]$ or $\hat{y}[t+1\|t]$	ES smoothing coefficient range (Inequality Ref. No.)	Use SC-Updater	Future info	Best model option (Input window size, RNN Cell, Hidden state dim., Loss function)
MES-LSTM	$l[t]$	Eq. (3)	X	None	Trend Predictor: (20, LSTM, 1024, RMSE)
Type 1	$\hat{y}[t+1\|t]$	Eq. (3)	X	None	Trend Predictor: (20, LSTM, 1024, RMSE)
Type 2	$\hat{y}[t+1\|t]$	Eq. (8)	X	None	Trend Predictor: (20, LSTM, 1024, RMSE)
AMES-RNN	$\hat{y}[t+1\|t]$	Eq. (8)	O	None	SC-Updater-Pedal: (10, LSTM, 1024, RMSE) SC-Updater-Steer: (10, GRU, 512, MSE) Trend Predictor: (20, GRU, 1024, RMSE)
AMES-RNN+	$\hat{y}[t+1\|t]$	Eq. (8)	O	Path curvature	SC-Updater-Pedal: (10, GRU, 1024, RMSE) SC-Updater-Steer: (15, LSTM, 512, RMSE) Trend Predictor: (20, LSTM, 1024, RMSE)
Type 1*	$y[t+1]$	Eq. (3) is not needed because finding $\lambda_0^{(i)}$ is not necessary.		None	Trend Predictor: (20, LSTM, 1024, RMSE)

TABLE 6. Comparison of 90th and 99th percentiles of MAE distribution.

Data Type	Pedal Position [-]		Steering Wheel Angle [deg]	
MAE Percentile	90%	99%	90%	99%
MES-LSTM	0.87	4.78	3.05	7.37
Type 1	0.74	4.10	2.51	5.93
Type 2	0.73	4.00	2.43	5.80
AMES-RNN	0.72	3.77	2.12	5.39
AMES-RNN+	0.67	3.23	1.37	3.56
Type 1*	0.61	3.11	2.11	5.02

TABLE 7. Statistics of one-step forecast errors for each ES model type.

ES Model Type	$\lvert\hat{y}[t+1\|t]-y[t+1]\rvert$ Statistics					
	Pedal Position [-]			Steering Wheel Angle [deg]		
	Mean	Std	Max	Mean	Std	Max
In Type 1	0.10	0.18	3.02	0.15	0.23	4.62
In Type 2	0.09	0.17	2.84	0.13	0.20	4.21
AMES	0.07	0.11	2.32	0.10	0.16	3.67

added as an offset to the trend component to be predicted by Trend Predictor. That is, the Trend Predictor needed to predict not only the future trend component but also the offset. On the other hand, since $\hat{y}[t+1|t]$ is used in Type 1, the Trend Predictor can relatively focus on the task of predicting the future trend component. For this reason, the prediction accuracy was improved. This is clearly demonstrated in the prediction result of the model denoted by "Type 1*" in the last row of Table 6. Type 1* is a non-causal version of Type 1, where the actual $y[t+1]$ is used instead of $\hat{y}[t+1|t]$ estimated by the ES model. When the actual $y[t+1]$ was used, the prediction accuracy was clearly improved, as the offset value of the future trend component to be predicted by the Trend Predictor always became zero.

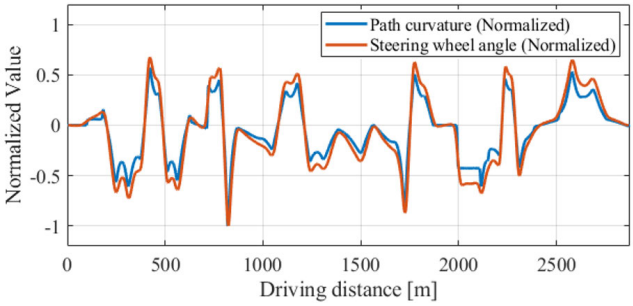


FIGURE 11. An example of path curvature and vehicle steering wheel angle data along the path (Note: Both data were normalized to clearly display their overall shapes).

Next, we can observe that the prediction accuracy was improved as the range (8) and the utilization of the SC-Updater were sequentially applied. This is because the smoothing coefficients that estimate $\hat{y}[t+1|t]$ more accurately were employed as each improvement was applied. Specifically, by replacing with (8), the optimal λ_0 was found in a wider range of smoothing coefficients. Additionally, with the addition of SC-Updater, the smoothing coefficient set that enabled the ES model to estimate $\hat{y}[t+1|t]$ more accurately than when using λ_0 was applied at each time step. The improvement in the estimation accuracy of $\hat{y}[t+1|t]$ due to these enhancements can be confirmed in Table 7. The table presents the prediction error statistics for each ES model type on the test data. As shown in the table, all statistical metrics for prediction error tended to decrease in the order of ES models used in Type 1, Type 2, and AMES-RNN. These indicate that the estimation accuracy of $y[t+1]$ was further improved.

Finally, the effect of using future-implying data was confirmed by comparing AMES-RNN with AMES-RNN+. By using the short-range forward road curvature profile data as additional input data, the prediction accuracy of both types of driving data was improved. In particular, the prediction for steering wheel angle showed significant improvement due to the relatively clear correlation between road curvature

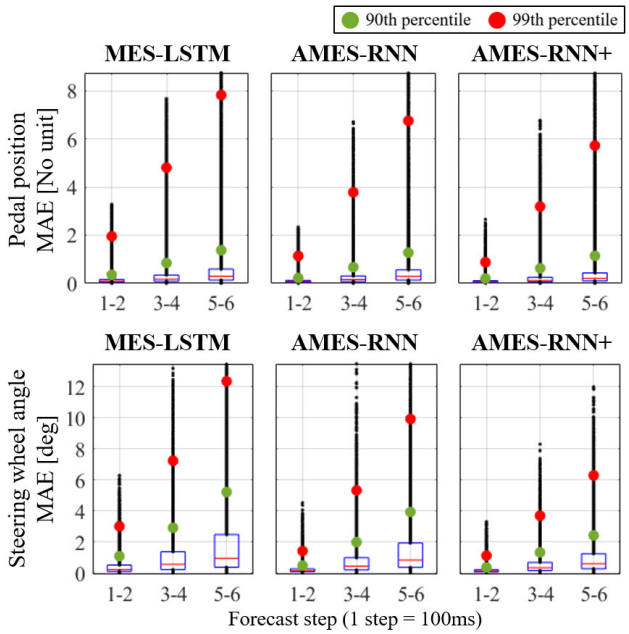


FIGURE 12. Box plot comparison of multistep prediction accuracy with results from all test tracks.

and steering wheel angle, as illustrated in Figure 11. This correlation contributes to reducing future uncertainty in predicting steering wheel angle data. Hence, the overall performance of the prediction model was improved. Through a comprehensive comparison of the accuracy test results for these five models, we confirmed that the proposed method can enhance the prediction accuracy.

Figure 12 shows the prediction accuracy by forecast step for the conventional MES-LSTM, the proposed AMES-RNN, and the case where future-implying data was additionally utilized. The comparison between the first and second results highlights that various improvements in the utilization of the ES model have led to a significant enhancement in future time series prediction accuracy compared to MES-LSTM. Also, in the third result, it is noticeable that the prediction accuracy of steering wheel angle was substantially improved by additionally using future-implying data.

D. COMPUTATIONAL EFFICIENCY

As online processing is another important factor in vehicle system implementation, this subsection presents the computational efficiency of AMES-RNN and AMES-RNN+ to evaluate whether the models can perform time series forecasting online. In this experiment, we measured the amount of memory consumed for model loading, FLOPs (FLoating point OPerations), and the average elapsed time required to make a 6-step forecast. Memory consumption was measured only once when the model was loaded, and the other two metrics were measured whenever a new observation was received using PyTorch Profiler. The observation consists of two driving data types (pedal position and steering wheel angle), which were measured at a rate of 10 Hz. All models

TABLE 8. Computational cost of the best models for each time series forecaster type.

Predictor	Memory usage [MB]	FLOPs	Run time [ms]
MES-LSTM	16.70	24.61×10^3	5.45
AMES-RNN	31.82	34.07×10^3	7.32
AMES-RNN+	34.09	34.12×10^3	12.24

Note: For AMES-RNN+, the running time is increased more significantly than the increase in FLOPs, because the overhead of concatenating future-implying data to the input data is not reflected in the FLOPs measurement by PyTorch Profiler.

were implemented on a typical computer with an Intel i7-11700K CPU, NVIDIA RTX 3080 Ti GPU, and 64GB of RAM. The results are summarized in Table 8. Values reported in the table is based on the inference phase of each time series forecaster, which is configured with the best model option in Table 5. For reference, the results of MES-LSTM are also presented to show the increase in computational cost of the AMES-RNN compared to MES-LSTM. Also, if the descriptions are applied to both AMES-RNN and AMES-RNN+, we collectively refer to them as "AMES-RNNs" for simplicity.

As the SC-Updaters are used, AMES-RNNs require more memory and computation than MES-LSTM. Nevertheless, each of the AMES-RNNs used in this experiment require less than 35 MB of memory, and perform only about 35×10^3 FLOPs for a single execution of a multistep forecast. Since most modern computing devices are equipped with at least several gigabytes (GB) of memory and can process more than 10^9 FLOPs per second, these computational resource requirements for performing online prediction can be easily met. Indeed, even for AMES-RNN+, which showed the longest execution time, the average computation time was less than 13 ms. This indicates that predictions can be sufficiently completed before receiving the next observation under the condition of 100 ms sampling interval. Along with these results, we demonstrated that AMES-RNNs can be utilized for online prediction.

VI. CONCLUSION

We propose AMES-RNN, a hybrid time series prediction model, for multi-step forecasting of multivariate time series data with non-seasonal and additive trend characteristics. The proposed model is based on MES-LSTM, and several improvements are applied to enhance prediction accuracy. First, we employ an RNN-based regression model to online estimate smoothing coefficients of the ES model that provides accurate predictions while maintaining the stability of the ES model. Second, we improve the input data preprocessing and output data postprocessing procedures for the RNN-based future trend component predictor. Finally, future-implying data is additionally provided as input data. To verify the effect of these enhancements on prediction performance,

we conducted multistep prediction accuracy tests on pedal position and steering wheel angle data acquired from the IPG CarMaker vehicle simulation software. As a result of analyzing the 90th and 99th percentiles of the prediction errors collected from tests where 6-step forecasts were performed at each time step, the prediction accuracy was improved by 23.0% and 32.4% for pedal position, and by 55.1% and 51.7% for steering wheel angle, respectively. Through these results, we confirmed that AMES-RNN provides superior prediction performance compared to MES-LSTM. In addition, we demonstrated that AMES-RNN requires only a small amount of computation and memory, which is sufficiently low for modern computing devices to handle, making it suitable for use as an online predictor in vehicles.

Many previous studies have demonstrated improvements in various vehicle functionalities by utilizing predicted future states of the vehicle. Considering these studies alongside the enhanced prediction accuracy presented in this paper, the proposed AMES-RNN has significant potential to improve the performance of such vehicle-related functions. Currently, we are conducting research to enhance vehicle control performance by utilizing AMES-RNN, and we are dedicated to demonstrating that it can contribute to enabling safer and more efficient vehicle operations. As part of our ongonig work, we will further validate the model performance using data collected from real-world driving environments, which contain various forms of noise, and explore other future-implying data (e.g., eye-gaze data, traffic interaction, etc.) that may improve the prediction accuracy under various driving conditions.

REFERENCES

[1] S. Haben, M. Voss, and W. Holderbaum, "Time series forecasting: Core concepts and definitions," in *Core Concepts and Methods in Load Forecasting*. Cham, Switzerland: Springer, 2023, pp. 55–66.

[2] X. Zhou, W. Bai, Y. Ren, Z. Yang, Z. Wang, B. Lo, and E. M. Yeatman, "An LSTM-based bilateral active estimation model for robotic teleoperation with varying time delay," in *Proc. Int. Conf. Adv. Robot. Mechatronics (ICARM)*, Guilin, China, Jul. 2022, pp. 725–730.

[3] S. B. Kamtam, Q. Lu, F. Bouali, O. C. L. Haas, and S. Birrell, "Network latency in teleoperation of connected and autonomous vehicles: A review of trends, challenges, and mitigation strategies," *Sensors*, vol. 24, no. 12, p. 3957, Jun. 2024.

[4] A. S. Bharatpur. (Jan. 2022). *A Literature Review on Time Series Forecasting Methods*. Accessed: Oct. 16, 2024. [Online]. Available: https://www.researchgate.net/publication/357786404

[5] L. Cascone, S. Sadiq, S. Ullah, S. Mirjalili, H. U. R. Siddiqui, and M. Umer, "Predicting household electric power consumption using multi-step time series with convolutional LSTM," *Big Data Res.*, vol. 31, Feb. 2023, Art. no. 100360.

[6] B. Gülmez, "Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm," *Exp. Syst. Appl.*, vol. 227, Oct. 2023, Art. no. 120346.

[7] Z. Xu, J. Yuan, L. Yu, G. Wang, and M. Zhu, "Machine learning-based traffic flow prediction and intelligent traffic management," *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 1, pp. 18–27, Mar. 2024.

[8] G. U. Yule, "On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers," *Philos. Trans. Roy. Soc. London. Ser. A, Containing Papers Math. Phys. Character*, vol. 226, pp. 267–298, Jan. 1927.

[9] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[10] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *Int. J. Forecasting*, vol. 20, no. 1, pp. 5–10, Jan. 2004.

[11] P. Farajiparvar, H. Ying, and A. Pandya, "A brief survey of telerobotic time delay mitigation," *Frontiers Robot. AI*, vol. 7, Dec. 2020, Art. no. 578805.

[12] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," *Information*, vol. 14, no. 11, p. 598, Nov. 2023.

[13] S. S. W. Fatima and A. Rahimi, "A review of time-series forecasting algorithms for industrial manufacturing systems," *Machines*, vol. 12, no. 6, p. 380, Jun. 2024.

[14] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, "An experimental review on deep learning architectures for time series forecasting," *Int. J. Neural Syst.*, vol. 31, no. 3, Mar. 2021, Art. no. 2130001.

[15] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Macau, China, Aug. 2023, pp. 6778–6786.

[16] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philos. Trans. A Math. Phys. Eng. Sci.*, vol. 379, no. 2194, Feb. 2021, Art. no. 20200209.

[17] E. Spiliotis, "Time series forecasting with statistical, machine learning, and deep learning methods: Past, present, and future," in *Forecasting With Artificial Intelligence: Theory and Applications*, 1st ed., Cham, Switzerland: Springer, 2023, pp. 49–75.

[18] T. Kim, J. Kim, Y. Tae, C. Park, J. -H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–25.

[19] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, New Orleans, LA, USA, Jan. 2022, pp. 9881–9893.

[20] Y. Chen and K. Wang, "Prediction of satellite time series data based on long short term memory-autoregressive integrated moving average model (LSTM-ARIMA)," in *Proc. IEEE 4th Int. Conf. Signal Image Process. (ICSIP)*, Wuxi, China, Jul. 2019, pp. 308–312.

[21] A. S. Azad, R. Sokkalingam, H. Daud, S. K. Adhikary, H. Khurshid, S. N. A. Mazlan, and M. B. A. Rabbani, "Water level prediction through hybrid SARIMA and ANN models based on time series analysis: Red hills reservoir case study," *Sustainability*, vol. 14, no. 3, p. 1843, Feb. 2022.

[22] X. Xu, X. Jin, D. Xiao, C. Ma, and S. C. Wong, "A hybrid autoregressive fractionally integrated moving average and nonlinear autoregressive neural network model for short-term traffic flow prediction," *J. Intell. Transp. Syst.*, vol. 27, no. 1, pp. 1–18, Jan. 2023.

[23] J. Guo, H. He, and C. Sun, "ARIMA-based road gradient and vehicle velocity prediction for hybrid electric vehicle energy management," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 5309–5320, Jun. 2019.

[24] Y. Jeong, S. Kim, and K. Yi, "Surround vehicle motion prediction using LSTM-RNN for motion planning of autonomous vehicles at multi-lane turn intersections," *IEEE Open J. Intell. Transp. Syst.*, vol. 1, pp. 2–14, 2020.

[25] A. Ezen-Can, "A comparison of LSTM and BERT for small corpus," 2020, *arXiv:2009.05451*.

[26] D. Wang and C. Chen, "Spatiotemporal self-attention-based LSTNet for multivariate time series prediction," *Int. J. Intell. Syst.*, vol. 2023, no. 1, pp. 1–16, Jan. 2023.

[27] B. Peng et al., "RWKV: Reinventing RNNs for the transformer era," 2023, *arXiv:2305.13048*.

[28] T. Choudhari and A. Maji, "Modeling driver's braking and steering behavior along horizontal curves of two-lane rural highways for ADAS applications," *Traffic Injury Prevention*, vol. 23, no. 7, pp. 404–409, Oct. 2022.

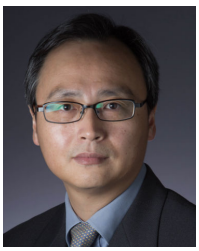
[29] H. Zhang and R. Fu, "A hybrid approach for turning intention prediction based on time series forecasting and deep learning," *Sensors*, vol. 20, no. 17, p. 4887, Aug. 2020.

[30] L. Cao, Y. Luo, Y. Wang, J. Chen, and Y. He, "Vehicle sideslip trajectory prediction based on time-series analysis and multi-physical model fusion," *J. Intell. Connected Vehicles*, vol. 6, no. 3, pp. 161–172, Sep. 2023.

[31] W. Jo, H. Park, and S. Lee, "V2X based lateral acceleration prediction for connected and automated vehicle," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2021, pp. 1675–1678.

[32] S. Alsanwy, H. Asadi, M. R. C. Qazani, M. Al-Ashmori, S. Mohamed, D. Nahavandi, A. A. Alqumsan, S. Al-Serri, S. M. Jalali, and S. Nahavandi, "Prediction of vehicle motion signals for motion simulators using long short-term memory networks," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Prague, Czech Republic, Oct. 2022, pp. 34–39.
[33] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003.
[34] M. Alizadeh, S. Rahimi, and J. Ma, "A hybrid ARIMA–WNN approach to model vehicle operating behavior and detect unhealthy states," *Exp. Syst. Appl.*, vol. 194, May 2022, Art. no. 116515.
[35] W. Wang, B. Ma, X. Guo, Y. Chen, and Y. Xu, "A hybrid ARIMA-LSTM model for short-term vehicle speed prediction," *Energies*, vol. 17, no. 15, p. 3736, Jul. 2024.
[36] R. N. Putri, M. Usman, and E. Virginia, "Modeling autoregressive integrated moving average (ARIMA) and forecasting of PT Unilever Indonesia Tbk share prices during the COVID-19 pandemic period," *J. Phys., Conf.*, vol. 1751, no. 1, Jan. 2021, Art. no. 012027.
[37] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *Int. J. Forecasting*, vol. 36, no. 1, pp. 75–85, Jan. 2020.
[38] T. Mathonsi and T. L. van Zyl, "A statistics and deep learning hybrid method for multivariate time series forecasting and mortality modeling," *Forecasting*, vol. 4, no. 1, pp. 1–25, Dec. 2021.
[39] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "ETSformer: Exponential smoothing transformers for time-series forecasting," 2022, *arXiv:2202.01381*.
[40] S. Zhang, R. Bai, R. He, Z. Meng, Y. Chang, Y. Zhi, and N. Sun, "Research on vehicle trajectory prediction methods in urban main road scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 16392–16408, Nov. 2024, doi: 10.1109/TITS.2024.3419037.
[41] X. Fan, F. Wang, D. Song, Y. Lu, and J. Liu, "GazMon: Eye gazing enabled driving behavior monitoring and prediction," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1420–1433, Apr. 2021.
[42] Z. Wang, R. Zheng, T. Kaizuka, and K. Nakano, "Relationship between gaze behavior and steering performance for driver–automation shared control: A driving simulator study," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 1, pp. 154–166, Mar. 2019.
[43] Y. Fu, H. Wang, and N. Virani, "Masked multi-step multivariate time series forecasting with future information," 2022, *arXiv:2209.14413*.
[44] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting With Exponential Smoothing: The State Space Approach*. Berlin, Germany: Springer, 2008.
[45] A. Mystakidis, E. Ntozi, K. Afentoulis, P. Koukaras, P. Gkaidatzis, D. Ioannidis, C. Tjortjis, and D. Tzovaras, "Energy generation forecasting: Elevating performance with machine and deep learning," *Computing*, vol. 105, no. 8, pp. 1623–1645, Aug. 2023.
[46] A. Jadon, A. Patil, and S. Jadon, "A comprehensive survey of regression-based loss functions for time series forecasting," in *Data Management, Analytics & Innovation*. Singapore: Springer, 2024, pp. 117–147.
[47] A. Novikov, I. Novikov, and A. Shevtsova, "Study of the impact of type and condition of the road surface on parameters of signalized intersection," *Transp. Res. Proc.*, vol. 36, pp. 548–555, Jan. 2018.
[48] National Weather Service. *Beaufort Wind Scale*. Accessed: Jan. 21, 2025. [Online]. Available: https://www.weather.gov/mfl/beaufort
[49] M. Pietruch, A. Wetula, and A. Mlyniec, "Influence of the accuracy and CAN frame period of the steering wheel angle sensor (SAS) on the trajectory of a steer-by-wire-equipped car," *IEEE Access*, vol. 10, pp. 106110–106116, 2022.
[50] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

JI HWAN SEO received the B.S. degree in transdisciplinary studies from Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, South Korea, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science. His research interests include motion planning and path tracking for autonomous driving platforms, as well as state prediction of vehicles, with a focus on improving the safety, reliability, and comfort.



KYOUNG-DAE KIM (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2011. He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Daegu Gyeongbuk Institute of Science and Technology (DGIST). Prior to joining DGIST, he was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Denver, USA. He was also a Postdoctoral Research Associate with the Department of Electrical and Computer Engineering, Texas A&M University, USA. His research interests include developing theories, tools, and software frameworks to improve the reliability and autonomy of cyber-physical systems and their application to real systems, such as smart transportation systems and collaborative robotic systems.