Article

Mixed Learning- and Model-Based Mass Estimation of Heavy Vehicles

Abdurrahman İşbitirici [1,2,*], Laura Giarré [2] and Paolo Falcone [2,3,*]

1 Department of Electrical, Electronic and Information Engineering, University of Bologna, 40126 Bologna, Italy
2 Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125 Modena, Italy; laura.giarre@unimore.it
3 Mechatronics Group, Department of Electrical Engineering, Chalmers University of Technology, 41296 Gothenburg, Sweden
* Correspondence: abdurrahm.isbitiric2@unibo.it (A.İ.); falcone@unimore.it (P.F.)

Abstract: This research utilized *long short-term memory (LSTM)* to oversee an RLS-based mass estimator based on longitudinal vehicle dynamics for heavy-duty vehicles (HDVs) instead of using the predefined rules. A multilayer LSTM network that analyzed parameters such as vehicle speed, longitudinal acceleration, engine torque, engine speed, and estimated mass from the RLS mass estimator was employed as the supervision method. The supervisory LSTM network was trained offline to recognize when the vehicle was operated so that the RLS estimator gave an estimate with the desired accuracy and the network was used as a reliability flag. High-fidelity simulation software was employed to collect data used to train and test the network. A threshold on the error percentage of the RLS mass estimator was used by the network to check the reliability of the algorithm. The preliminary findings indicate that the reliability of the RLS mass estimator could be predicted by using the LSTM network.

Keywords: mass estimation; recursive least squares; long short-term memory

check for updates

Citation: İşbitirici, A.; Giarré, L.; Falcone, P. Mixed Learning- and Model-Based Mass Estimation of Heavy Vehicles. *Vehicles* **2024**, *6*, 765–780. https://doi.org/10.3390/vehicles6020036

Academic Editor: Mohammed Chadli

Received: 14 March 2024
Revised: 19 April 2024
Accepted: 27 April 2024
Published: 30 April 2024

1. Introduction

Accurate knowledge of a vehicle's mass is crucial for HDVs, as it plays a crucial role in improving the *energy efficiency* and assuring *safety*. This significance becomes apparent when considering the optimization of powertrain operations, such as automatic gear shifting, where algorithms can enhance the performance by leveraging information about the mass of the vehicle [1]. Moreover, data concerning mass is essential in offering customized suggestions to truck drivers, enabling them to modify their driving technique according to the present vehicle mass and road conditions, thereby enhancing the safety and effectiveness in general. The estimation of the braking distance can be derived from the current mass information [2]. In addition, the combination of mass information and inertia is used by active safety systems to reduce the likelihood of a rollover [3]. The mass information can be utilized to ascertain the sequencing of trucks within a platoon [4]. For example, the leading position in a platoon might be assigned to the heaviest truck, considering its longer braking distance and the potential rear-end collision risk with lighter trucks while following them closely. Conversely, when traveling uphill, it may be advantageous to arrange heavier trucks in the rear of the platoon in order to prevent lighter vehicles from being unnecessarily slowed down. Mass estimation plays a diverse role in the context of buses, assisting in many activities, such as optimizing ventilation settings, resource planning, and bus schedules regarding the number of passengers [5].

Learning-based or *model-based* approaches can be applied to estimate the mass or load of HDVs. Learning-based methods can be easily adapted for various vehicles and diverse road conditions. They can successfully capture nonlinear and complex patterns. In [6], a neural network (NN) that had two hidden layers with eight neurons in the first layer and four neurons in the second layer was used to estimate the mass and road grade of a heavy vehicle. Six different road scenarios, each with 10 km of data for eight different masses, were generated for this purpose. Vehicle velocity, longitudinal acceleration, and engine torque at their previous time step values, as well as the previous time step mass and road grade estimations with uniform noise, were used as inputs to the network for training. In [7], a deep neural network (DNN) was proposed to estimate the trailer mass. Fifteen fully connected layers, with a total of 1051 neurons, were used to estimate the mass. The authors in [8] employed a random forest algorithm to categorize the weight of e-scooter users. The preference for categorization over regression arises due to the scarcity of accessible data. In our previous study [9], we employed an LSTM network with double hidden layers to estimate a truck mass with a 0–4-ton load range. Additionally, ref. [10] proposed a data-driven mass estimator based on the bi-directional gated recurrent unit, claiming a mean absolute percentage error of less than 1%. Although learning-based methods are beneficial for accurately estimating mass by learning from real-world data, it must be highlighted that these methods heavily rely on the quality and availability of data. Also, generalization cannot be done for unseen conditions. High-variety and high-quality data are necessary for effective training. Another issue with learning-based methods is that difficulties in understanding the reasoning behind the estimation make learning-based methods less interpretable. Therefore, their trustworthiness may be questioned.

Despite the dependency on vehicle modeling and road dynamics, which involves challenging processes, such as obtaining vehicle parameters and inertia values and having limited adaptability to changing conditions, model-based mass estimation is the most preferred method due to its physical interpretability, reduced data dependency, and robustness in cases where an accurate model is used. The automobile industry has made substantial contributions to vehicle mass estimation. A mass estimate based on RLS was developed in [11]. This estimator utilizes longitudinal dynamics and metrics, including engine torque, longitudinal acceleration, brake pressures, gearbox data, and wheel speeds. The first mass estimation is calculated based on data from seat belt sensors and fuel level information, and the RLS method can be reset when any car door is opened. In [12], the RLS algorithm is applied to estimate vehicle mass by evaluating fluctuations in the height of the center of gravity of the vehicle, as measured through lidar technology. The approach outlined in [13] unfolds in three steps: the last vehicle mass estimation in the volatile memory upon engine initiation, subsequent estimations based on longitudinal dynamics during straight-line motion, and a refined mass calculation during acceleration and deceleration events after the vehicle reaches the set speed to mitigate errors arising from acceleration and speed measurement noise. Ref. [14] presented an approach to estimate the mass of a vehicle using readings of vertical acceleration from inertial sensors. This technique utilizes the known mass of the truck while it is empty to estimate the mass of the load using an RLS approach, assuming that longitudinal and lateral accelerations are minimal. Lastly, in [4], a specialized method for estimating the mass of vehicle platoons is presented. This approach depends on assessing the disparity in mass between neighboring vehicles by utilizing longitudinal dynamics and sharing data between trucks, such as torque, braking, and gap information.

Prevalently employed model-based algorithms for vehicle mass estimation commonly utilize the RLS method, as evidenced by numerous studies [15–20] in the literature. In [15], an RLS estimation algorithm was employed to concurrently estimate the mass, drag coefficient, and rolling resistance of a vehicle. This estimation depends on the longitudinal dynamics of the vehicle. This approach involves first estimating the gradient of the road and then estimating the mass by using a second-order Butterworth low-pass filter to filter the road grade, vehicle speed, engine torque, and acceleration. In [16], an RLS mass estimation algorithm was employed that utilizes data derived from measured signals that are meticulously extracted through a supervisory algorithm designed to identify maneuvers characterized by inertial dynamics, such as acceleration maneuvers. In [17], the need to use RLS with multiple forgetting factors for estimating the mass and road grade is highlighted. It was recognized that not using separate forgetting factors for the mass and road grade leads to unsatisfactory outcomes. A nonlinear road grade and mass estimator is introduced in [18]. This estimate is initialized with initial estimates of the mass and road grade, which are obtained using an adaptive least squares approach. This study assumed that both parameters remain constant in the first stage. Then, the varying road gradient is computed in the second stage while considering this assumption. The RLS mass estimator described in [19] is based on the principle of longitudinal and roll vehicle dynamics. The RLS technique is proficient in simultaneously estimating the mass and road grade when the latter is not directly accessible. The technique for RLS mass estimation, as described in [20], improves the mass estimate by following a predefined set of criteria. The requirements include many conditions: the longitudinal acceleration must exceed 0.1 m/s^2, the engine torque percentage should not be less than 40% or more than 90%, the lateral acceleration must be below 5 m/s^2, and the absolute engine torque gradient should not exceed 1000 Nm/s. Furthermore, the mass estimation is revised only when no braking is present and certain gears are engaged. The criteria are carefully developed to correspond with the vehicle dynamics. The accuracy of the mass estimate is improved by using an accurate vehicle model for the RLS estimation procedure. However, we highlight that the formulation of such rules, while conducive to accuracy, may end up with too much time-consuming effort. Indeed, the implementation of an RLS-based mass estimator, as documented in the literature, necessitates the integration of supplementary logic. This additional logic plays a pivotal role in facilitating mass updates, especially when the vehicle model exhibits a high degree of accuracy. Nevertheless, the process of refining and adjusting such reasoning might be arduous and unfeasible.

The standard RLS algorithm is used with a growing window of data. Ref. [21] provided a guide on the real-time implementation of RLS. Old data are retained but the impact of them is diminished by using a forgetting factor. Instead of the growing window approach, ref. [22] proposed a sliding window RLS algorithm that incorporates a time-varying regularization term. The oldest data are removed when the new data are available, and thus, the algorithm operates on a finite window. In [23], the effect of the regularization-induced bias on parameter estimation accuracy with RLS is explored. In instances characterized by a deficiency in persistence, the incorporation of a regularization term is a customary practice. Nonetheless, it is imperative to acknowledge that this augmentation, while addressing the aforementioned inadequacy, introduces a significant perturbation to the regressor. In [24], variable-rate forgetting is introduced as an alternative to constant-rate forgetting, ensuring convergence to true parameters, even in cases where noise is uncorrelated with the regressor under persistent excitation. Ref. [25] claimed that if excitation is not persistent, although parameter estimates converge, they may not converge to their true values, and the error covariance state might diverge. Variable-direction forgetting is proposed in [25] to prevent the divergence of covariance. Necessary conditions for the global asymptotic stability of RLS are examined in [26] since persistence excitation is a sufficient but not necessary condition. Ref. [27] conducted stability and robustness analysis for various RLS extensions. Lastly, ref. [28] introduced two extensions of RLS with exponential resetting and cyclic resetting.

The paper [29] presents a hybrid methodology that combines RLS with a machine learning framework based on fuzzy logic. The machine learning component of this hybrid system operates through an NN with two hidden layers characterized by 1024 and 256 neurons. The fusion of a longitudinal-vehicle-dynamics-based RLS and NN involves the application of fuzzy logic to weigh the estimates generated by both methodologies. The weighting depends on signals such as vehicle speed, longitudinal acceleration, and acceleration rate. Moreover, another weight is given to the former mass estimation, which acts as a backup when both the NN and RLS techniques demonstrate unreliability. Thus, three weights of fuzzy logic are set based on the defined rules of the authors such that the weight for dynamic-model-based mass estimation should be increased at higher speeds since the algorithm converges, as opposed to at lower speeds.

LSTM is used in a variety of domains, such as natural language processing, time series prediction, and speech recognition. In [30], convex-based LSTMs were used for dynamical systems identification. A multivariate brake pressure estimation algorithm based on LSTM was proposed in [31]. Also, cutting-edge anomaly detection methods focusing on LSTMs were surveyed in [32].

Recursive least squares (RLS) is widely used in the literature to estimate vehicle mass. However, obtaining predefined rules stating when RLS should be used is challenging and time consuming. Also, these rules may not be easily adaptable for different vehicles. Therefore, this study delved into the utilization of LSTM networks to capture long-term dependencies for reliability classification of an RLS-based mass estimator. The primary objective was to classify whether mass estimation is reliable or not by using LSTM to supervise the RLS estimation.

The structure of this paper is arranged in the following manner. Section 2 consists of three parts: in Section 2.1, an RLS mass estimation algorithm is presented; Section 2.2 explains how to determine the network architecture and training procedure; and in Section 2.3, an LSTM-based network is explained to determine whether RLS is reliable or not. The results are shown in Section 3, starting with data collection in Section 3.1, followed by the presentation of RLS reliability based on certain conditions in Section 3.2. Lastly, the conclusions are written in Section 4.

2. Mixed-Model- and Learning-Based Mass Estimator

2.1. Recursive Least Squares Mass Estimator

To present the RLS-based mass estimation, we need to present the vehicle modeling. The design of RLS is based on the longitudinal dynamics depicted in Figure 1. Longitudinal dynamics is written based on the second law of Newton. Hence, the mass of the vehicle, denoted as m, can be represented by Equation (1).

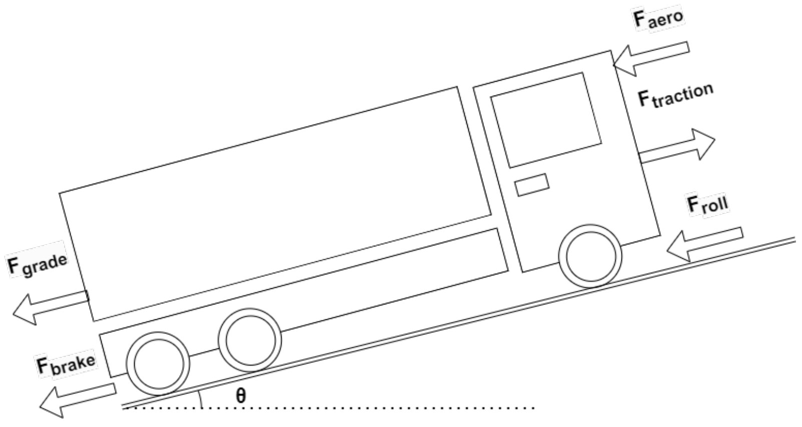

Figure 1. Longitudinal vehicle dynamics.

$$m = \frac{F_{traction} - F_{brake} - F_{aero} - F_{roll} - F_{grade}}{a}. \tag{1}$$

The traction force $F_{traction}$ can be calculated as in (2), where the engine torque T_{engine} is typically available from the engine control unit, along with the engine speed ω. The current gear g_r is also typically available from the vehicle CAN bus. The parameters of the equivalent inertia J_{eq}, the final gear ratio g_{final}, and the nominal wheel radius r_{wheel} are available from the vehicle design.

$$F_{traction} = \frac{T_{engine} - J_{eq}\dot{\omega}}{\frac{r_{wheel}}{g_r g_{final}}}. \tag{2}$$

In (1), F_{brake} is the braking force, which is not considered in this study because it needs a more detailed model that includes a combination of a foundation brake, retarder, and engine brake, as highlighted in [33]. The aerodynamic drag force F_{aero} can be calculated as in (3), where the air density ρ can be estimated by using look-up tables and maps based on onboard sensors, such as temperature, pressure, and mass air flow sensors. The aerodynamic drag coefficient c_d is obtained by wind tunnel testing or determined by computational fluid dynamics (CFD) simulations. The frontal area of the truck A_f is available from the truck design. The longitudinal acceleration a and velocity v can be obtained by onboard sensors, such as accelerometers, wheel speed sensors, or GPS systems. Although the wind speed v_{wind} and direction can be measured by a wind vane and anemometer on a weather station, this is not possible at road level.

$$F_{aero} = 0.5\rho c_d A_f (v + v_{wind})^2. \tag{3}$$

Since the rolling resistance force F_{roll} and road grade force F_{grade} are directly dependent on the mass of the truck, they are not included in the regressor. Instead, their effect on the acceleration is added to the output. The rolling resistance coefficient μ can be obtained by testing or using established data, such as from tire manufacturers or research studies. Inclinometers, tilt sensors, or GPS-based elevation data can be used to obtain road gradient angles θ.

$$F_{roll} = \mu mgcos(\theta), \tag{4}$$

$$F_{grade} = mgsin(\theta). \tag{5}$$

We show how the mass estimation algorithm shown in Figure 2 can be built upon an RLS estimation algorithm and an LSTM. RLS is an online algorithm for recursively estimating time-varying parameters. The classical approach is described in many books on adaptive control, signal processing, or identification, such as [34–36].

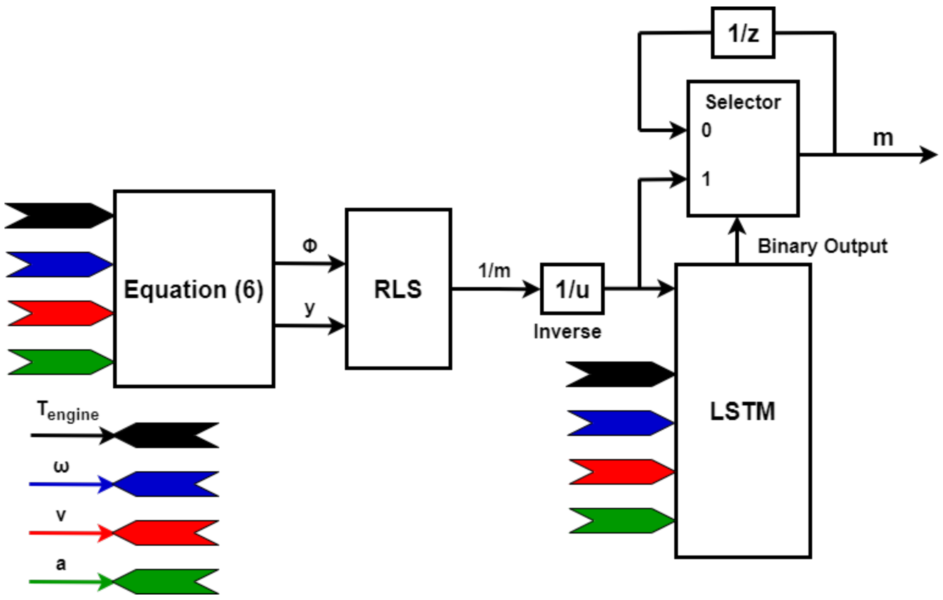

Figure 2. The block diagram of the proposed mixed-model- and learning-based mass estimation algorithm.

The regressor ϕ can be built based on the models (2) and (3) and the set of measurements made available by the onboard sensing and communication devices. Meanwhile, y is obtained based on models (4) and (5), in addition to the acceleration measurement of an inertial measurement unit (IMU). Equation (1) can be expanded by using (2)–(5) and rewritten as in (6).

$$\Theta = \frac{1}{m},\ y = a + \mu g cos(\theta) + g sin(\theta),\ \phi = F_{traction} - F_{aero}. \tag{6}$$

After the regressor ϕ and the output y are computed, they are used to find a sliding window RLS estimator with a shorter window length while achieving convergence. By trial and error, an RLS estimator was used with a two-minute finite window length with a 0.01 s sampling time.

2.2. Network Architecture and Training Procedure

The amplitudes of the input signals directly affect the training of an NN. *Feature normalization* is crucial to mitigate issues that arise from unbalanced feature magnitudes that could impede the training process. In addition to this, normalization can be done for the outputs of the layers, which is called layer normalization.

To compute the parameters of (the weights **W** and biases **b**) during training, a set of *hyperparameters* [37], which are configuration settings that can not be learned from the data during training, needs to be determined. Hyperparameter selection is a significant step to make the learning process possible and faster. For NNs, some hyperparameters include the following:

- *NN type*—The architecture is determined based on the problem by choosing an NN type (feedforward, recurrent, convolutional, etc.). If the task requires spatial data, then a convolutional NN (CNN) may be preferred, whereas an RNN is preferred for sequential data. If there are tabular data, then a shallow NN or DNN might be preferred based on the complexity of the problem. In the context of time series modeling, it is worth noting that DNNs can also be employed for dynamical systems. However, a key distinction arises in how they handle sequential data compared with RNNs. When DNNs are applied to a time series, past values of the same feature are treated as separate inputs because DNNs do not have memories. Consequently, each of these inputs has its own set of weights, leading to an increase in the overall number of parameters. This arises from the fact that distinct weights are assigned to each input, contributing to a potential growth in model complexity. In contrast, RNNs demonstrate a more generic approach because the network handles temporal dependencies using cyclic connections. Hence, the weights for various time steps of a feature do not change, and thus, the same weights are used. Thus, an RNN can capture temporal dependencies without enormously escalating the number of parameters. Nevertheless, since a vanilla RNN is vulnerable to gradient-vanishing problems, a special type of RNN named LSTM was preferred in this study.
- *Number of hidden layers*—After the NN type is settled on, the number of hidden layers to be used needs to be determined. Although deeper architectures are better at handling complex dependencies, they are computationally more expensive, and they may result in overfitting. Deep networks may need more data to generalize better. On the other hand, shallow networks may lead to underfitting, where the model is unable to capture the dynamics.
- *Number of units*—After deciding how many hidden layers to use, the quantity of units in the hidden layers is determined. As the number of neurons or cells increases, the computational complexity increases, and the network is prone to overfitting, whereas fewer neurons may cause the network to be unable to capture the complexity.
- *Activation function*—Activation functions are determined based on the use case (a regression or classification problem) and on the NN type. Sigmoid is preferred for binary classification problems, whereas softmax is used for multi-class classification. ReLU is the most preferred activation function for regression problems with a DNN. Hyperbolic tangent is used for LSTMs.

- *Training-validation-testing ratio*—The dataset is separated into three parts: training data, validation data, and test data.
- *Loss function*—The loss function is chosen based on the problem. Loss functions, such as the mean square error (MSE) or mean absolute error (MAE), are used for regression problems, whereas cross-entropy can be used for classification problems.
- *Batch size*—If processing the whole training dataset at once is not possible due to the hardware constraints, dataset size, and complexity of the NN, then the dataset can be divided into smaller datasets named mini-batches. Therefore, the batch size is the number of training instances that are trained in one iteration. The batch size should evenly divide the total number of training data. During each iteration of training, the parameters of a mini-batch are updated.
- *Epoch number*—The epoch number represents how many times the learning algorithm is applied to the entire training dataset. It is crucial to choose a sufficiently large maximum epoch number to ensure thorough training of the network. Meanwhile, the number of iterations in one epoch is obtained by dividing the number of training datasets by the batch size.
- *Early stopping criteria*—If the cost function does not diminish for consecutive predefined times, the training is terminated without considering the maximum number of epochs.
- *Optimizer*—Stochastic gradient descent (SGD), root-mean-square propagation (RMSprop), and adaptive moment estimation (ADAM) are commonly used optimizers to minimize the loss function during the training.
- *Learning rate*—If the learning rate is chosen as being too high, the loss function may diverge. If it is not too high but high, the loss function may not converge to the optimal solution. A smaller learning rate leads to a longer training time, requiring more epochs. The selection of the learning rate should be contingent upon the optimization technique employed for training the NN. The learning rate can either remain constant or decrease after a specified number of iterations.
- *Dropout*—Dropout is a regularization method used to mitigate overfitting by randomly eliminating selected units throughout each iteration of training [38].
- *Weight and bias initialization*—Various initialization methods, such as Glorot (Xavier) [39], He [40], orthogonal, uniform, Gaussian, or zero or one initialization, can be used to start the training.
- *Sequence length*—While longer sequences can capture dynamics better, they can cause a more challenging training process. The sequence length should be chosen based on the specific dataset. Although the sequence length may vary from simulation to simulation, a constant simulation time may be preferred if it is possible in order not to need sequence padding or truncation.

The construction of an NN entails a repetitive process that includes separate stages:

1. *Design*—The network configuration is established by selecting the type of NN, specifying the number of layers, choosing the number of units for each layer, and determining the activation functions. Units can be neurons, kernels, or cells based on the NN type. The activation functions are chosen based on the NN type and based on use cases, such as classification and regression. Meanwhile, different NN types can be used in combination, based on the problem. The parameters $\mathbf{W}$ and $\mathbf{b}$ are defined by choosing the network architecture. If the network architecture is designed such that not enough weights and biases are defined or if the number of data trained is insufficient, then the training accuracy will be lower and *underfitting* occurs since the network cannot fully learn. In this case, the amount of high-quality data or the quantity of layers or units can be increased. On the other hand, if more than the required parameters are defined in a network, the network also unintentionally learns the noise during the training. This phenomenon is called *overfitting*.
2. *Learning*—Parameters are obtained by minimizing the difference between the predicted output and the actual measurements based on a cost function. Hyperparameters, such as the training-validation-test ratio, dropout, epoch number, weight

initialization, number of batches, and early stopping criteria, are crucial for ensuring a swift and accurate learning process. In order to detect and prevent overfitting, the validation data set is used. Therefore, validation is also done during training, and overfitting can be observed by assessing the learned model on a separate validation dataset. In the scenario of overfitting, the discrepancy between the predicted output and the actual output, as measured on the validation dataset, is significantly greater than the error observed on the training dataset. To address the overfitting, it is recommended to consider using a network architecture with a reduced number of parameters during the initial design phase. Additionally, to mitigate the issue of overfitting, the technique of *dropout* might be employed.

3. *Testing*—The loss function is mainly used to determine the weights and biases for training data, whereas performance metrics are used to evaluate the performance after the model is trained. MAE, MSE, root MSE (RMSE), or R2 can be used as regression metrics, whereas accuracy, precision, recall, or F1 score can be used as classification metrics. Consequently, the acquired model is evaluated using novel data that are distinct from the training or validation datasets. The predicted output from the trained model is compared with the true system output to evaluate the model.

For classification problems, performance metrics, such as accuracy or precision, can be obtained by using the confusion matrix shown in Table 1 [41].

Table 1. Confusion matrix.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True positive	False negative
	Negative	False positive	True negative

Accuracy is the number of correctly predicted classes out of the total number of samples, as shown in (7). If the classes are imbalanced, only considering the accuracy might give false outcomes [42]. *Precision* is the number of correctly predicted positives out of the predicted positives, as shown in (8). In order to avoid false positives, a precision metric can be used.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \quad (7)$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (8)$$

2.3. LSTM Supervisor

The LSTM network was first presented in [43]. Nonlinear hidden neurons are replaced with LSTM cells to handle the gradient vanishing within RNNs. But, the main contribution is done in [44] by adding the forget gate to make the LSTM cell learn resetting when necessary, in addition to other gates. An LSTM cell structure is demonstrated in Figure 3.

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f), \quad (9)$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i), \quad (10)$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o). \quad (11)$$

The forget gate is represented by f, as in (9); the input gate is represented by i, as in (10); and the output gate is represented by o, as in (11). These three gates are used for erasing, writing, and reading purposes. The logistic function σ is chosen for gate activation, whereas hyperbolic tangent is used for input and output activation functions. W_{*x} and W_{*h} are the input and recurrent weights, respectively.

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c), \tag{12}$$

$$h_t = o_t \odot tanh(c_t). \tag{13}$$

The cell state c_t and hidden state h_t are shown in (12) and (13). The cell state deletes some info from the previous cell state and writes new cell info, and the hidden state reads some info from the cell, as seen in Figure 3. Element-wise multiplication is denoted by $\odot$.

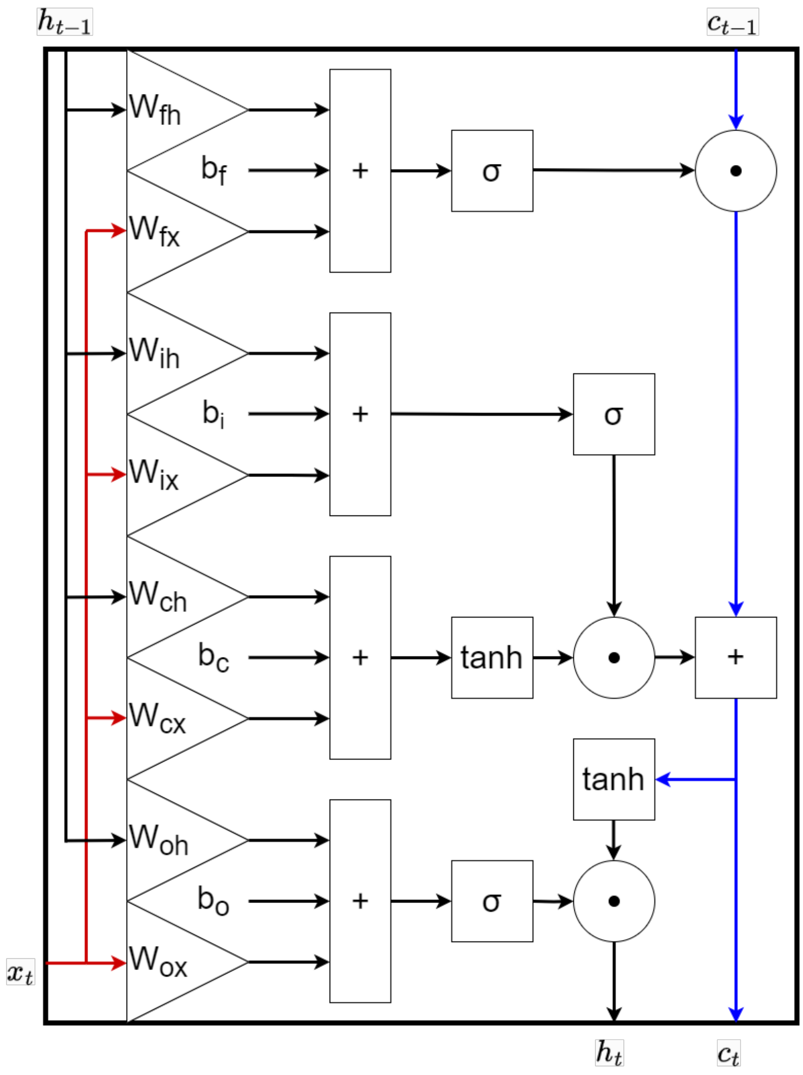

Figure 3. LSTM cell [45].

Determining the reliability of an RLS estimator is a binary classification problem. Since the data are time-based and estimation is done in each time interval, this problem is a sequence-to-sequence classification problem.

To check whether the RLS mass estimator was reliable or not, a two-layered LSTM network was designed. Vehicle speed, engine speed, engine torque, longitudinal acceleration, and estimated mass were chosen as the inputs of the LSTM, as shown in Figure 2, in order to train the network. Preprocessing was done by normalizing all inputs with min–max normalization. Labeled outputs were binary: if the mass estimation was not within acceptable thresholds, the output value was set to false. If it was reliable, the output was true. Since there were five features, the size of the normalized input vector of current time $X_N(t)$ was 5 × 1. The sequence length was set to 12,000 since the horizon window length of the RLS was 120 s with a 0.01 s sampling time. Therefore, each piece of data consisted of a 5 × 12,000 array. As shown in (9)–(13), values of gates and information about hidden and cell states are calculated at every time interval.

Ten LSTM cells were used in both hidden layers. Outputs of the first layer are normalized by layer normalization by computing the mean and standard deviation [46]. Then, dropout with a 0.2 probability was used to prevent overfitting, as shown in Figure 4. A dense layer was used after the second LSTM layer in order to transmit information from cells to the output at each time step. Sigmoid or softmax with two classes can be used as the output layer to determine the reliability of the RLS mass estimator. Therefore, approximately 1500 parameters were trained.

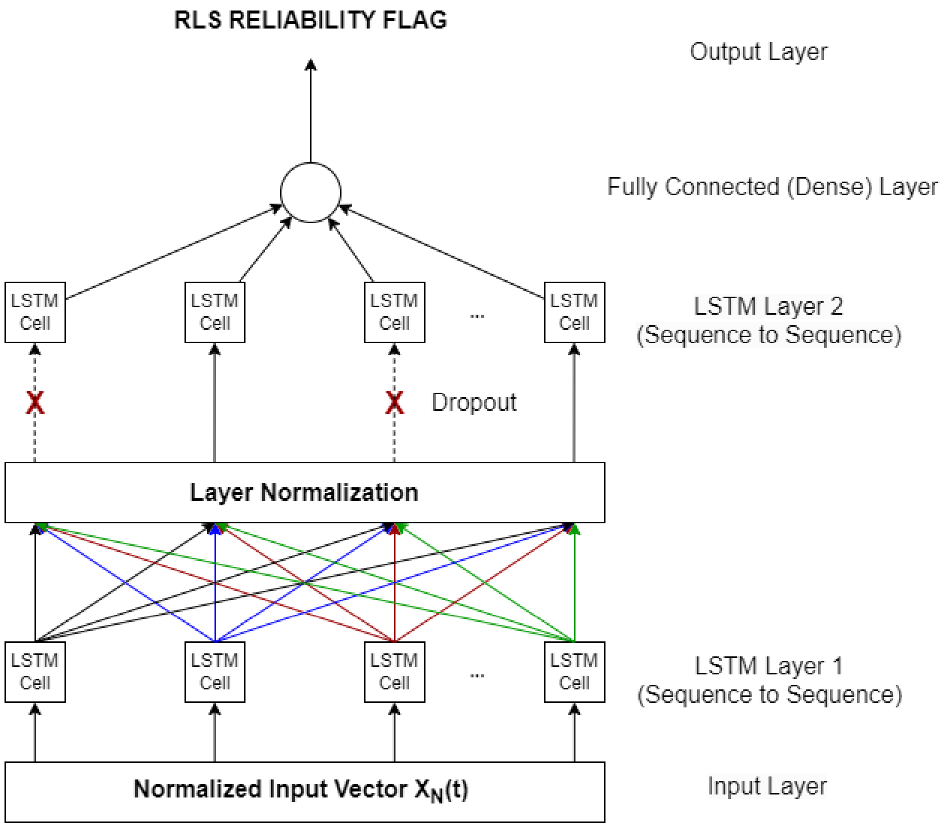

Figure 4. LSTM-based RLS supervisor.

Since the sequence length was chosen to be 2 min, the 450 min dataset (explained in Section 3.1) was split into 216 instances. Seven out of nine data were used for training, one out of nine data were used for validation, and remaining data were used for testing.

The epoch number was chosen as 5000 by a process of trial and error to adequately train the network. An ADAM optimizer was preferred, with the initial learning rate set to 0.002. The learning rate decayed by 1% after every 100 epochs. The batch size was 168 to train the entire training data at once; therefore, the number of iterations was equal to the number of epochs.

3. Results

3.1. Data Generation

Truckmaker simulation software was used to generate synthetic data. Since excitation is necessary for RLS to accurately estimate mass, the accelerator pedal was frequently pressed and released. The Volvo 4 × 2 truck model in Truckmaker 11.0 with a fixed load was chosen to execute some maneuvers. The data obtained included the accelerator pedal position, engine speed, engine torque, vehicle speed, and longitudinal acceleration.

The available parameters that were necessary for the longitudinal dynamics were taken from Truckmaker.

The following criteria were used for the data collection:

- *Without braking*—RLS mass estimator was not used during braking since braking dynamics are much more complex and less accurate than traction dynamics.
- *Engaged clutch*—Data were generated while the clutch was fully engaged so that the engine was always engaged with the transmission. This criterion was used to minimize the effect of different drivers and transients. Otherwise, the RLS mass estimator would be available for larger intervals; however, the estimation accuracy would diminish.
- *Fixed gear*—The data samples were for the highest gear in order to reduce the number of data and the number of necessary parameters because data collection for all gears would not increase the mass estimation accuracy, although it would make RLS available for longer periods.
- *Limited speed*—Data samples were generated based on specified vehicle speed intervals. RLS was assumed to be used on highways; therefore, the speed limit was chosen based on the highest gear, where the minimum speed was defined as 60 km/h, whereas the maximum speed of the vehicle was 95 km/h.
- *Without steering*—Data were generated while driving on a straight road because only longitudinal vehicle dynamics were considered.
- *Flat roads*—Downhill or uphill scenarios were not included in order to decrease the complexity.
- *Without wind*—Wind effects were not included in order to have a more accurate model. For simulation purposes, it could be added, but in reality, it is not known how to obtain wind information at the road level.

The unladen mass of the truck was 7 tons. The load interval was chosen between 0 and 4 tons, with increments of 0.5 tons. Therefore, nine simulation datasets were collected with 3000 s of simulation time and a 100 Hz sampling frequency. Therefore, a total of 450 min of data were collected. Since convergence had not occurred yet, the first two minutes of each simulation dataset were removed.

3.2. Reliability Analysis

The LSTM-based reliability classification algorithm for the RLS mass estimator, which was designed as written in Section 2.3, was trained using the Deep Learning toolbox of MatLab R2022a with the training and validation datasets. An NVIDIA GA100 [A100 PCIe 40GB] on Lyra at Unimore Data Centre in Italy was used for computation.

If the mass estimation error was less than 3.5%, then RLS was assumed to be reliable; otherwise, it was unreliable. A finite horizon length has an effect on choosing the error percentage threshold because RLS is less reliable when the window length is chosen to be shorter. Another important factor is having a balance between two classes of outputs while choosing the error percentage threshold.

A total of 168 training data and 24 validation data instances, each with a 2 min sequence length, were trained. The training and validation accuracy is shown in Figure 5. Weights and biases were obtained based on the minimum loss function, while the reliability prediction was undertaken on the test data. The predictions of 6 out of 24 test instances are shown in Figure 6.

The confusion matrix for 24 test instances, with each having 12,000 data points, is shown in Table 2. The reliability accuracy of the LSTM on the entire test data was 92.24%. Nevertheless, accuracy may be replaced with precision metrics. If the LSTM predicts a false negative, it can be tolerated since the estimator can use the previous mass estimate. However, the LSTM must not predict false positives. Therefore, precision, which was a more suitable metric for this problem, was calculated as 91.60%.

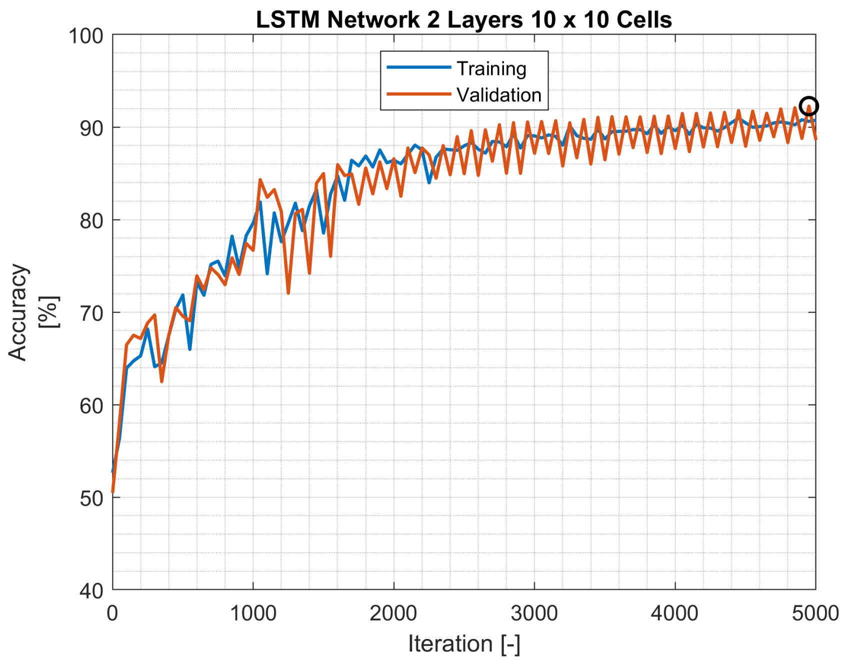

Figure 5. Training and validation accuracy.

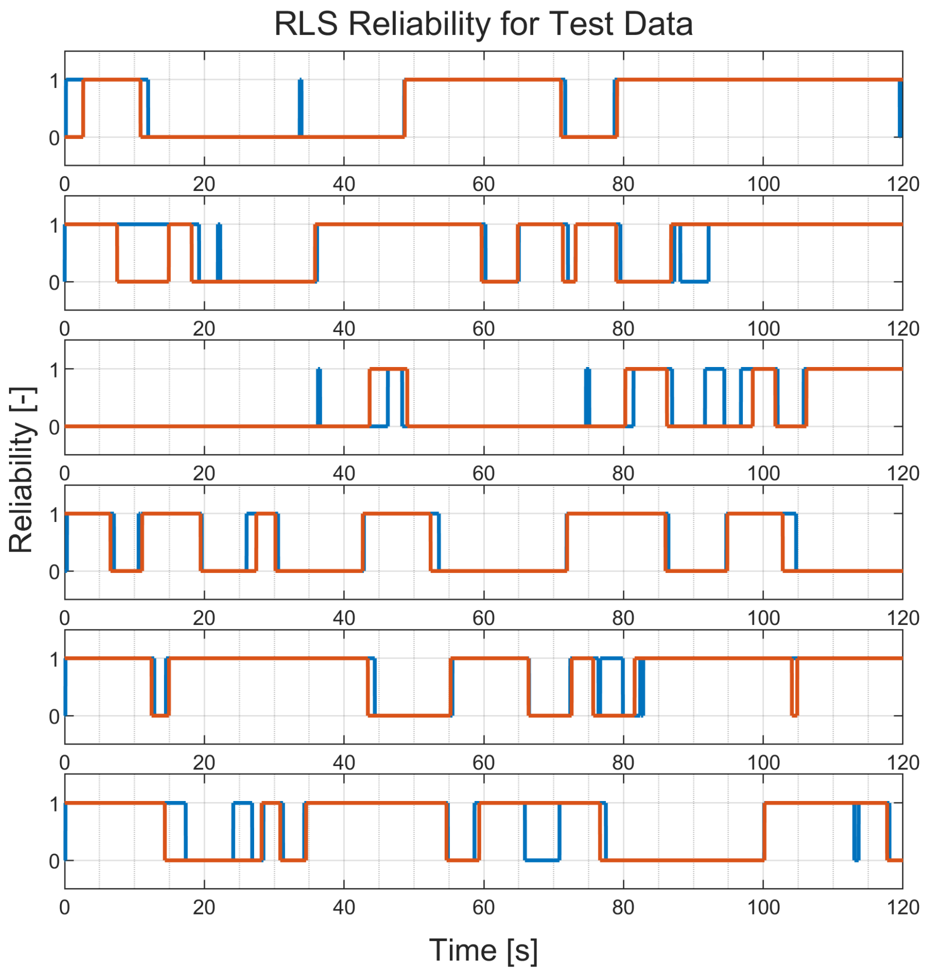

Figure 6. RLS reliability of 6 various test data with 2 min lengths. The blue line represents the LSTM result, whereas the orange line represents the real RLS reliability.

Table 2. Confusion matrix for RLS reliability.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	129,057	10,509
	Negative	11,833	136,601

In Figure 7, we report the results where, instead of using synthetically generated maneuvers, a driving cycle that is a worldwide harmonized light vehicle test cycle (WLTC) class 2 was used for a truck with an 11-ton mass. RLS with LSTM supervision almost always estimated the mass more accurately. There were several reasons why the error percentage was higher in this graph. First, the vehicle model was not fully known. Second, it was not expected from the algorithm to perform well for the gears except the highest gear since the LSTM was only trained for the highest gear. Thus, only gear 8 should be considered. Another reason was that WLTC not only has acceleration parts but also includes braking parts. Also, the data were not always persistently excited. Lastly, the finite horizon length was 120 s, and thus, the convergence took time, and a gear change or braking affected the results. In some intervals, the error percentage was around 6%, as seen between 1240 s and 1300 s since a 120 s convergence was used and braking was negligible in this period.

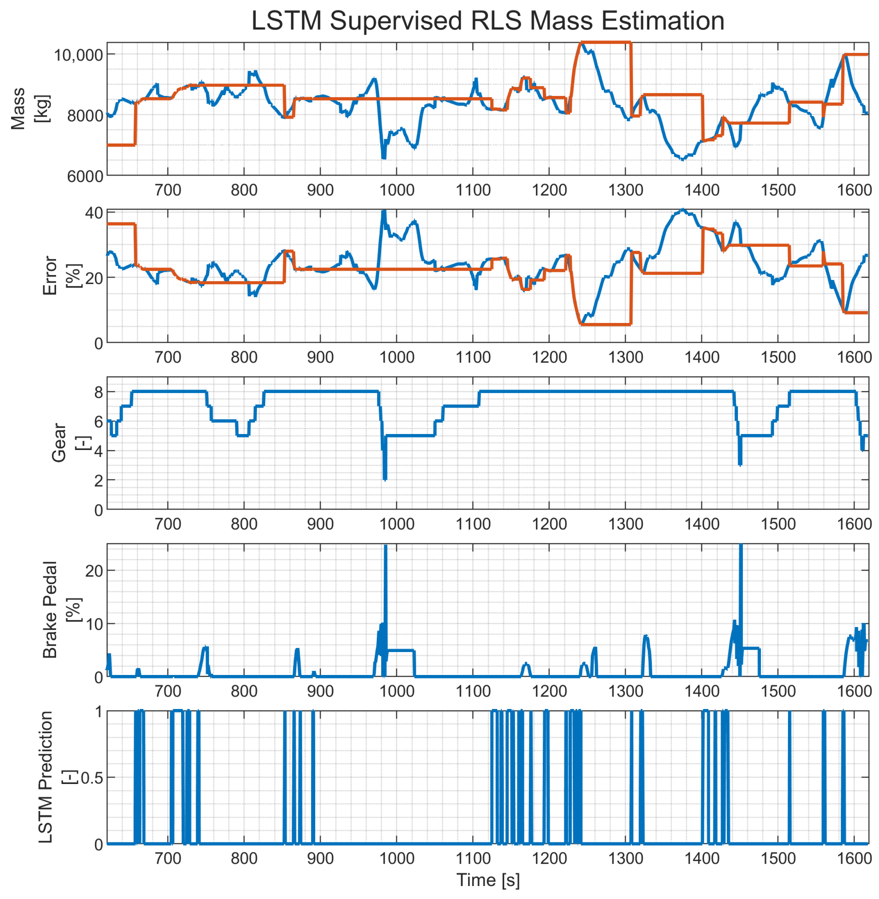

Figure 7. RLS reliability on WLTC data. The blue line represents the RLS results, whereas the orange line represents the LSTM-supervised RLS results for the first two sublots.

4. Conclusions and Future Study

In conclusion, this study demonstrated the efficiency of employing LSTM as a supervisory mechanism for RLS in estimating the mass of an HDV. Multilayer LSTM network integration as a reliability flag is successful in assessing the dependability of the RLS mass estimator. The error percentage threshold to categorize whether the RLS mass estimator is either reliable or unreliable was chosen based on the driving cycle and the window length of the RLS to have balanced data to classify. Our preliminary findings indicate that with an accuracy exceeding 90%, this methodology enabled a precise forecast of the reliability of the RLS mass estimator. However, the limitations of this study were as follows: The clutch was always assumed to be engaged and only the highest gear was used; therefore, the training data were collected at specified vehicle speed intervals. Braking scenarios were excluded since more investigation was necessary about the brake model and measurements of brake-related signals. Only straight-driving and flat-road cases were considered. The wind effects were not included.

In the future, this study can be extended by collecting the truck data for various gears, including downhill and uphill scenarios. Therefore, how various road gradients affect LSTM supervision will be seen. However, more data are necessary in this case. Also, the network complexity may increase when classifying the data more accurately because the number of inputs increases. Therefore, more computational resources might be necessary. Also, this algorithm can be tested for electric or fuel cell vehicles by changing the inputs of the networks based on the chosen vehicle. If more accurate braking and retarder information can be obtained, the network can also include braking parts. Instead of using realistic simulation data, real truck data can be used.

Author Contributions: Conceptualization, L.G.; methodology, P.F.; software and data curation, A.İ. All authors read and agreed to the published version of the manuscript.

Funding: Only the work of Abdurrahman İşbitirici is supported by the Ministry of National Education of the Republic of Türkiye.

Data Availability Statement: The data presented in this study are available if requested from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: The authors would like to thank IPG Automotive for providing the TruckMaker license. The authors would also like to thank the University of Modena and Reggio Emilia for providing hardware.

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

References

1. Xu, C.; Geyer, S.; Fathy, H.K. Formulation and comparison of two real-time predictive gear shift algorithms for connected/automated heavy-duty vehicles. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7498–7510. [CrossRef]
2. Chen, Y.L.; Shen, K.Y.; Wang, S.C. Forward collision warning system considering both time-to-collision and safety braking distance. In Proceedings of the 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), Melbourne, Australia, 19–21 June 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 972–977.
3. Kober, W.; Hirschberg, W. On-board payload identification for commercial vehicles. In Proceedings of the 2006 IEEE International Conference on Mechatronics, Budapest, Hungary, 3–5 July 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 144–149.
4. Switkes, J.P.; Erlien, S.M.; Schuh, A.B. Applications for Using Mass Estimations for Vehicles. U.S. Patent 20220229446-A1, 21 July 2022.
5. Ritter, A. Optimal Control of Battery-Assisted Trolley Buses. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2021.
6. Torabi, S.; Wahde, M.; Hartono, P. Road grade and vehicle mass estimation for heavy-duty vehicles using feedforward neural networks. In Proceedings of the 2019 4th international conference on intelligent transportation engineering (ICITE), Singapore, 5–7 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 316–321.
7. Korayem, A.H.; Khajepour, A.; Fidan, B. Trailer mass estimation using system model-based and machine learning approaches. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12536–12546. [CrossRef]
8. Leoni, J.; Strada, S.; Tanelli, M.; Savaresi, S.M. Real Time Passenger Mass Estimation for e-scooters. In Proceedings of the 2023 American Control Conference (ACC), San Diego, CA, USA, 31 May–2 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1741–1746.

9. İşbitirici, A.; Giarré, L.; Xu, W.; Falcone, P. LSTM-Based Virtual Load Sensor for Heavy-Duty Vehicles. *Sensors* **2024**, *24*, 226. [CrossRef] [PubMed]
10. Zhang, H.; Yang, Z.; Shen, J.; Long, Z.; Xiong, H. Dynamic mass estimation framework for autonomous vehicle system via bidirectional gated recurrent unit. *IET Control Theory Appl.* 2023, *Early View*.
11. Mittal, A.; Fairgrieve, A. Vehicle Mass Estimation. U.S. Patent 20180245966-A1, 30 August 2018.
12. Rezaeian, A.; Li, D. Vehicle Center of Gravity Height Detection and Vehicle Mass Detection Using Light Detection and Ranging Point Cloud Data. U.S. Patent 20220144289-A1, 12 May 2022.
13. Huang, X. Method for Real-Time Mass Estimation of a Vehicle System. U.S. Patent 20190186985-A1, 20 June 2019.
14. Jundt, O.; Juhasz, G.; Weis, R.; Skrabak, A. System and Method for Identifying a Change in Load of a Commercial Vehicle. U.S. Patent 20220041172-A1, 10 February 2022.
15. Bae, H.S.; Ryu, J.; Gerdes, J.C. Road grade and vehicle parameter estimation for longitudinal control using GPS. In Proceedings of the IEEE Conference on Intelligent Transportation Systems, Oakland CA, USA, 25–29 August 2001; pp. 25–29.
16. Fathy, H.K.; Kang, D.; Stein, J.L. Online vehicle mass estimation using recursive least squares and supervisory data extraction. In Proceedings of the 2008 American Control Conference, Seattle, WA, USA, 11–13 June 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1842–1848.
17. Vahidi, A.; Stefanopoulou, A.; Peng, H. Recursive least squares with forgetting for online estimation of vehicle mass and road grade: Theory and experiments. *Veh. Syst. Dyn.* **2005**, *43*, 31–55. [CrossRef]
18. McIntyre, M.L.; Ghotikar, T.J.; Vahidi, A.; Song, X.; Dawson, D.M. A two-stage Lyapunov-based estimator for estimation of vehicle mass and road grade. *IEEE Trans. Veh. Technol.* **2009**, *58*, 3177–3185. [CrossRef]
19. Kim, D.; Choi, S.B.; Oh, J. Integrated vehicle mass estimation using longitudinal and roll dynamics. In Proceedings of the 2012 12th International Conference on Control, Automation and Systems, Jeju Island, Republic of Korea, 17–21 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 862–867.
20. Paulsson, E.; Åsman, L. Vehicle Mass and Road Grade Estimation using Recursive Least Squares. Master's Thesis, Lund University, Lund, Sweden, 2016.
21. Islam, S.A.U.; Bernstein, D.S. Recursive least squares for real-time implementation [lecture notes]. *IEEE Control Syst. Mag.* **2019**, *39*, 82–85. [CrossRef]
22. Hoagg, J.B.; Ali, A.A.; Mossberg, M.; Bernstein, D.S. Sliding window recursive quadratic optimization with variable regularization. In Proceedings of the 2011 American Control Conference, San Francisco, CA, USA, 29 June–1 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 3275–3280.
23. Lai, B.; Islam, S.A.U.; Bernstein, D.S. Regularization-induced bias and consistency in recursive least squares. In Proceedings of the 2021 American Control Conference (ACC), 26–28 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 3987–3992.
24. Bruce, A.L.; Goel, A.; Bernstein, D.S. Convergence and consistency of recursive least squares with variable-rate forgetting. *Automatica* **2020**, *119*, 109052. [CrossRef]
25. Goel, A.; Bruce, A.L.; Bernstein, D.S. Recursive least squares with variable-direction forgetting: Compensating for the loss of persistency [lecture notes]. *IEEE Control Syst. Mag.* **2020**, *40*, 80–102. [CrossRef]
26. Bruce, A.L.; Goel, A.; Bernstein, D.S. Necessary and sufficient regressor conditions for the global asymptotic stability of recursive least squares. *Syst. Control Lett.* **2021**, *157*, 105005. [CrossRef]
27. Lai, B.; Bernstein, D.S. Generalized Forgetting Recursive Least Squares: Stability and Robustness Guarantees. *arXiv* **2023**, arXiv:2308.04259.
28. Lai, B.; Bernstein, D.S. Exponential Resetting and Cyclic Resetting Recursive Least Squares. *IEEE Control Syst. Lett.* **2022**, *7*, 985–990. [CrossRef]
29. Yu, Z.; Hou, X.; Leng, B.; Huang, Y. Mass estimation method for intelligent vehicles based on fusion of machine learning and vehicle dynamic model. *Auton. Intell. Syst.* **2022**, *2*, 4. [CrossRef]
30. Wang, Y. A new concept using LSTM Neural Networks for dynamic system identification. In Proceedings of the 2017 American Control Conference (ACC), Seattle, WA, USA, 24–26 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 5324–5329.
31. Xing, Y.; Lv, C. Dynamic state estimation for the advanced brake system of electric vehicles by using deep recurrent neural networks. *IEEE Trans. Ind. Electron.* **2019**, *67*, 9536–9547. [CrossRef]
32. Lindemann, B.; Maschler, B.; Sahlab, N.; Weyrich, M. A survey on anomaly detection for technical systems using LSTM networks. *Comput. Ind.* **2021**, *131*, 103498. [CrossRef]
33. Torabi, S. Fuel-Efficient Driving Strategies. Ph.D. Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2020.
34. Ljung, L. *System Identification: Theory for the User*; Prentice Hall: Upper Saddle River, NJ, USA, 1999.
35. Söderström, T.; Stoica, P. *System Identification*; Prentice Hall International: Upper Saddle River, NJ, USA, 2001.
36. Aström, K.J.; Wittenmark, B. *Adaptive Control*; Addison Wesley: Boston, MA, USA, 1995.
37. Chowdhury, K. *10 Hyperparameters to Keep an Eye on for Your LSTM Model—and Other Tips*; Geek Culture: Singapore, 2023.
38. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
39. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.

40. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
41. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
42. Scikit. Metrics and Scoring: Quantifying the Quality of Predictions. 2024. Available online: https://scikit-learn.org/stable/modules/model_evaluation.html (accessed on 19 January 2024).
43. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
44. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef] [PubMed]
45. Ng, A. Sequence Models Complete Course. 2021. Available online: https://www.youtube.com/watch?v=S7oA5C43Rbc (accessed on 29 April 2024).
46. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.