

An Optimal Knowledge Retention Framework for Continual Learning in Data Stream Scenarios

by

Arvin Hosseinzadeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2025

© Arvin Hosseinzadeh 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Qing Zhao
Professor, Dept. Electrical and Computer Engineering,
University of Alberta

Supervisors: Amir Khajepour
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo

Shojaeddin Chenouri
Professor, Dept. of Statistics and Actuarial Sciences,
University of Waterloo

Internal Member: William Melek
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo

Internal-External Member: Liquan Diao
Associate Professor, Dept. of Statistics and Actuarial Sciences,
University of Waterloo

Internal-External Member: Sirisha Rambhatla
Assistant Professor, Dept. of Management Science and Engineering,
University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the field of time series and data stream analysis, neural networks (NNs) have demonstrated excellent performance in predicting current and future states of dynamic systems. However, forgetting a previously learned information by NN when training the model on new data can be a significant challenge in having a reliable prediction, a problem that is known as catastrophic forgetting (CF) in neural networks. Unfortunately, retraining the model with both historical and new data is often impractical due to computational complexity and storage constraints, particularly in large-scale applications. One of the most prominent examples is automotive systems, where dynamic environments, such as changing road conditions or driving scenarios, require continuously updating the existing information based on new data.

The main objective of this thesis is to propose a continual learning method that can efficiently train a neural network model on newly collected data while preserving previously acquired information. A novel framework based on memory-based continual learning approaches is developed, consisting of two critical tasks: optimal sampling of the old data to store in memory, and optimization. First, the proposed method aims to identify the most relevant and informative memories for old dataset, which are then contributed in future learning to preserve the previously learned information. The proposed method is developed in both univariate and multivariate time series prediction scenarios. Second, a proper optimization technique is used in each training epoch to minimize the loss function by modifying the network parameters, ensuring that NN is capable of successfully integrating new input while maintaining historical information.

Additionally, a hybrid state estimation framework is introduced, leveraging the selected memory points to detect distribution shifts in real-time within the incoming data stream. When the estimator detects unfamiliar patterns that may degrade the predictive performance of the neural network, it adaptively switches to a model-based estimator to ensure robust and reliable estimation under the newly encountered conditions. A variety of neural network models and architectures are explored and compared to provide a comprehensive analysis and to evaluate their effectiveness in state estimation tasks. Furthermore, uncertainty analysis is conducted using conformal prediction, enabling quantification of the neural network’s predictive uncertainty after training on each task and comparison to a conventional batch learning baseline.

The proposed framework is applied to both univariate and multivariate scenarios for estimating vehicle longitudinal and lateral velocities, incorporating new driving maneuvers into the previously trained neural network model. Experimental datasets comprise of

sensor measurements from an electric Equinox vehicle. The effectiveness of the method is evaluated by examining the performance of the model in training on new information as well as the impact of forgetting on previously acquired knowledge as new tasks are incrementally introduced. The findings of this study suggest that the developed continual learning framework is capable of efficiently training the model on new data while preserving the prediction accuracy on previous data. The time efficiency of the proposed method is an important advantage, as it enables the neural network to adapt to new tasks quickly without a significant computational overhead.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Professor Amir Khajepour and Professor Shojaeddin Chenouri, for their invaluable support, guidance, and patience throughout my PhD journey. Their insight, encouragement, and calm approach to problem-solving have been instrumental in both my academic and personal growth. I am truly grateful for their unwavering commitment, generosity with their time, and their belief in my work.

I would like to extend my deepest gratitude to my thesis internal and internal-external committee members: Professors William Melek, Sirisha Rambhatla, and Liqun Diao. Thank you for your time, thoughtful feedback, and guidance throughout my Ph.D. journey. Their expertise, insightful questions, and encouragement during my comprehensive exam, helped strengthen the quality and impact of this work. I sincerely thank Professor Qing Zhao for agreeing to serve as the external committee member of this thesis.

I am also sincerely thankful to Professor Mohammad Pirani for his thoughtful advice and constructive feedback during my doctoral studies. Undoubtedly, without the mentorship and supervision of my supervisors and advisors, this work would not have been possible.

I gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for providing the essential financial and educational support that enabled this research.

My deepest appreciation goes to my parents, Faramarz and Zahra, and to my beloved brother, Pouya. Their unwavering love, sacrifices, and constant encouragement have been the foundation of all my achievements. Their incredible support throughout my life has shaped who I am today, and I am forever indebted to them.

I would also like to thank Professor. Joseph (Chen) Sun, Dr. Reza Valliollahi Mehrizi and Dr. Ladan Khoshnevisan for their valuable advice, encouragement, and the many meaningful discussions that contributed to my academic progress. I would also like to thank the technician in the MVS laboratory, Jeff Graansma for helping me in road experimnts and data extraction.

Finally, I would like to extend my heartfelt thanks to my wonderful friends, Mohammadreza Ghorbani, Farid Mafishourestani, Pouya Panahandeh, Mehdi Zabihi, Mobin Khamooshi, Ehsan Mohammadbagher, Ali Gharamohammadi, Yukun Lu, Ziaming Zhong, and Keqi Su, for their friendship, support, and for making this journey more enjoyable. Their companionship, laughter, and encouragement have meant a great deal to me.

To all those who have supported me along the way, thank you.

Dedication

This thesis is dedicated to my beloved parents.

Table of Contents

Examining Committee	ii
Author’s Declaration	iii
Abstract	iv
Acknowledgements	vi
Dedication	vii
List of Figures	xii
List of Tables	xvi
List of Abbreviations	xviii
Nomenclature	xx
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Thesis Outline	4

2	Literature Review	7
2.1	Introduction	7
2.2	Local Regression Methods	9
2.3	Continual Learning Methods	10
2.3.1	Regularization-based Methods	10
2.3.2	Architectural-based Methods	11
2.3.3	Generative Replay	12
2.3.4	Memory-based Methods	12
2.3.5	Continual Learning Applications to Time Series Forecasting	13
2.4	A Review of Advanced Machine Learning Methods in Vehicle State Estimation	14
2.5	Summary	15
3	Representative Selection and Continual Learning in Time Series Regression Tasks	16
3.1	Introduction	16
3.2	Formulating Continual Learning in Time Series Regression	18
3.3	Optimization with Average Gradient of Episodic Memory	19
3.4	Capturing Key Information of a Function via Change Point Detection	22
3.4.1	Representative Selection in Time Series Dataset	25
3.5	Evaluation and Testing	28
3.5.1	Dataset Description	28
3.5.2	Phase 1: Initial Training (Longitudinal Velocity)	32
3.5.3	Phase 2: Continual Learning (Longitudinal Velocity)	35
3.5.4	Phase 1: Initial Learning (Lateral Velocity)	40
3.5.5	Phase 2: Continual Learning (Lateral Velocity)	41
3.5.6	On the sensitivity of the number of memory points in estimation error	42
3.5.7	Comparison of the performance after introducing multiple domains	44
3.6	Summary	49

4	Developing the Continual Learning Framework in Multi-output Time Series Regression Tasks	52
4.1	Introduction	52
4.2	Formulating the Problem in Multi-output Regression	53
4.3	Refined A-GEM Optimization algorithm	54
4.4	Multi-output Representative Selection in time series tasks	55
4.5	Evaluation and Testing	60
4.5.1	Phase 1: Initial Training	64
4.5.2	Phase 2: Continual Learning	68
4.5.3	EM-ReSeleCT v.s. A-GEM: Performance on memory loss	73
4.5.4	Uncertainty Quantification with Conformal Prediction	73
4.5.5	On the input sensitivity analysis and feature selection	75
4.6	Summary	80
5	Developing an adaptive Real-time Hybrid Vehicle State Estimation	82
5.1	Introduction	82
5.2	Problem Formulation	82
5.3	Hybrid State Estimator	85
5.3.1	Representative Selection and Continual Learning	86
5.4	Evaluation and Testing	88
5.4.1	Initial training and estimation results on the dry road	89
5.4.2	Estimation under Distribution Shift: Wet and Icy Roads	90
5.4.3	Continual Learning	92
5.5	A Comparative Study on the Choice of Neural Network Model for State Estimation	94
5.6	Summary	99
6	Conclusions and Future Work	100
6.1	Conclusions	100
6.2	Future Work	101

List of Figures

1.1	General scheme of the proposed method.	5
2.1	General categorizes of existing methods in lifelong learning scenarios. . . .	8
3.1	The scheme of a memory-based continual learning framework.	17
3.2	The position of the two gradients in parameters space; (a): The scenario where the inner product between the two gradients is positive. (b): The scenario where the inner product between the two gradients is negative, along with the new projected gradient.	21
3.3	(a): An example of desired locations to store in memory. (b): change point detection in each randomly drawn interval in a time series data.	23
3.4	Proper time sequence to capture in memory.	26
3.5	Step-by-step procedure of the proposed method.	29
3.6	Step-by-step procedure of the evaluation.	30
3.7	Test vehicle used for experimental analysis.	30
3.8	Structure of an LSTM cell with input, forget, and output gates.	33
3.9	Estimation of V_x on the two test maneuvers after the initial training (D_1).	36
3.10	Estimation of V_x on the new data, prior to training the model on.	37
3.11	Estimation of V_x on the new dataset (D_2), after training the model on using ReSeleCT.	38
3.12	Estimation of V_x on the two test maneuvers after introducing the new dataset (D_2).	39

3.13	Comparison of the proposed method with other CL strategies in estimating V_x on testing data after introducing the new dataset (D_2) to the model.	40
3.14	Lateral velocity estimation on two test maneuvers after the initial training (D_1).	42
3.15	Estimation result on the new task, prior to training the model on.	43
3.16	Longitudinal velocity estimation on two test maneuvers after the introducing the first new domain (D_2).	45
3.17	Comparison of the proposed method with other CL strategies in estimating V_y after introducing the new dataset (D_2) to the model.	46
3.18	Sensitivity of the number of memory points on MAE for estimation of V_x after introducing the new domain using ReSeleCT.	47
3.19	Estimation error of Vx for initial dataset after introducing each maneuver.	48
3.20	Estimation error of Vy for initial dataset after introducing each maneuver.	48
3.21	Training time for each model on Vx after introducing new maneuvers.	50
3.22	Training time for each model on Vy after introducing new maneuvers.	50
4.1	The position of the two gradients in different scenarios. (a): When the inner product is positive. (b): When the inner product is negative and the magnitude of the new gradient is larger. (c): When the inner product is negative, and the magnitude of the new gradient is smaller.	56
4.2	Representative selection on each dimension separately.	57
4.3	Multi-output representative selection by filtering based on error fit.	59
4.4	Step-by-step procedure of the proposed method (EM-ReSeleCT).	62
4.5	The flowchart of evaluation process of different methods in continual learning scenarios.	63
4.6	Structure of the Transformer model used in this experimental study.	66
4.7	Estimation of V_x and V_y on the test data (domain D_1), after the initial training.	68
4.8	Estimation of V_x and V_y on the new data (domain D_2), prior to continual learning.	69
4.9	V_x and V_y estimations on the test maneuver in domain D_2 , after incorporating the new dataset (in D_2).	71

4.10	V_x and V_y estimations on the test maneuvers (domain D_1), after incorporating the new dataset (in D_2).	72
4.11	Evaluation of the proposed method against other continual learning (CL) strategies for estimating V_x and V_y on test data after incorporating the new dataset (D_2) into the model.	73
4.12	Tracking the loss error over each training epoch in continual learning with EM-ReSeleCT and A-GEM.	74
4.13	Uncertainty analysis of different models by number of selected memory points, on the old (test) dataset. a and b: coverage rate and interval width vs the number of memory points on V_x , respectively. c and d: coverage rate and interval width vs the number of memory points on V_y , respectively. . .	76
4.14	Sensitivity of the outputs to the change in each input signal.	78
4.15	Sensitivity of memory point selection to threshold and distance parameters in EM-ReSeleCT; (a): Under fixed threshold. (b): Under fixed distance. . .	81
5.1	The proposed procedure for adaptive state estimation.	85
5.2	Input signals to the encoder and decoder layers at each time step.	90
5.3	V_x estimation on the test maneuvers (domain D_1), after training the model on the initial dataset (dry road) with no distribution shift.	91
5.4	V_y estimation on the test maneuvers (domain D_1), after training the model on the initial dataset (dry road), with no distribution shift.	91
5.5	V_x estimation on under the distribution shift (wet road), using NN, model, and hybrid-based estimators.	92
5.6	V_x estimation on under the distribution shift (icy road), using NN, model, and hybrid-based estimators.	93
5.7	V_y estimation on under the distribution shift (icy road), using NN, model, and hybrid-based estimators.	93
5.8	V_x estimation on the wet road, after continual learning of the model using the new data distribution.	94
5.9	V_x and V_y estimation on the icy road, after continual learning of the model using the new data distribution.	95
5.10	V_x and V_y estimation on the dry road (initial dataset), after continual learning of the model using the new data distributions.	95

5.11 NARX Architecture: (a) Series-parallel structure. (b) Parallel structure. . .	96
5.12 Comparison of the performance on V_x and V_y estimation of different NN-based models, on test 1.	98
5.13 Comparison of the performance on V_x and V_y estimation of different NN-based models, on test 2	98

List of Tables

3.1	Description of the dataset for estimation of longitudinal and lateral velocities.	31
3.2	The specifications of hyperparameters of the neural network for estimation of V_x .	34
3.3	Estimation results of V_x on training dataset after initial training.	35
3.4	Estimation results of V_x on test dataset after initial training (D_1).	35
3.5	Estimation results of V_x on the new dataset (D_2) after training the model on.	37
3.6	Estimation Results (MAE) of V_x on training and testing maneuvers (D_1) after introducing the new dataset (D_2) to the model.	38
3.7	Hyperparameters of the neural network for estimating lateral velocity.	41
3.8	Estimation results of V_y on training dataset after initial training.	41
3.9	Estimation results of V_y on test datasets after initial training (D_1).	41
3.10	V_y estimation results on new maneuver (D_2) after training the model on.	43
3.11	Estimation Results (MAE) of V_y on Training and testing maneuvers after introducing the new maneuver (D_2) to the model.	44
3.12	Memory Points vs. MAE of estimation of V_x for initial dataset and testing sets	44
3.13	Description of the extended dataset for estimation of longitudinal and lateral velocities under multiple addition of domains.	49
4.1	Dataset used for evaluation of different methods to estimate vehicle velocity in a continual learning scenario.	64
4.2	Hyperparameter setting of the Transformer model for estimation of V_x and V_y .	67

4.3	Estimation results of V_x and V_y on initial and test dataset after initial training (D_1).	68
4.4	MAE and training time of V_x and V_y estimation on the new maneuver (D_2) following model training.	70
4.5	MAE of V_x and V_y estimation on training and testing maneuvers (in D_1 and D_2) after introducing the new maneuver (D_2).	72
4.6	Sensitivity scores for each input signal with respect to V_x and V_y	77
4.7	Estimation error (MAE) for V_x and V_y after sequentially removing the least sensitive input signals.	79
5.1	Hyperparameter configurations for NARX, LSTM, and Transformer models	97
5.2	Training times and estimation error (MAE) of the three models on two test sets.	99

List of Abbreviations

A-GEM	Average Gradient of Episodic Memory
A-GEM	Average Gradient Episodic Memory
AR	Autoregressive
ARIMA	auto regressive moving averag
CF	Catastrophic Forgetting
CL	Continual Learning
DNN	Deep Neural Network
EKF	Extended Kalman Filter
EM-ReSeleCT	Efficient Multivariate ReSeleCT
EMA	Exponential Moving Average
EV	Electric Vehicle
EWC	Elastic Weight Consolidation
GEM	Gradient of Episodic Memory
GLR	Generalized Likelihood Ratio
GNN	Graph Neural Networks
GPR	Gaussian Process Regression
GPS	Global Positioning System
IMU	Inertial Measurement Unit
L-GPR	Local Gaussian Process Regression
LSTM	Long Short-Term Memory
LWL	Locally Weighted Learning
MAE	Mean Absolute Error
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
MSE	Mean Squared Error
NARX	Nonlinear AutoRegressive model with eXogenous inputs

NLP	Natural Language Processing
NN	Neural Network
NOT	Narrowest Over Threshold
PNN	Progressive Neural Network
RAN	Resource Allocating Networks
ReSeleCT	Representative Selection for Continual Learning in Time series
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SI	Synaptic Intelligence
SLFN	Single Hidden Layer Feedforward Networks
SoC	State of Charge
SoC	State of Charge
SoH	State of Health
sSIC	strengthened Schwarz Information Criterion
SVM	Support Vector Machine
UKF	Unscented Kalman Filter

Nomenclature

AA_y	Vehicle angular acceleration
AA_z	Vehicle yaw acceleration
AV_x	Vehicle roll rate
A_t	Number of attention heads in the Transformer model
A_z	Vehicle vertical acceleration
D_i^*	Input-output pair representing the memory for dataset D_i
$D_i, \dots$	Data domains representing different road or condition datasets
E	Error between the two GLR values in the multi-output scenario
F_i	Tire force at the i th wheel
$H_i(D)$	History set of the i th sample with length d from dataset D
$S_{i,j}$	Sensitivity of the j th output with respect to the i th input
V_x	Vehicle Longitudinal velocity at CG
V_y	Vehicle Lateral velocity at CG
$\mathfrak{R}_{(s,e]}^c$	Generalized Likelihood Ratio value for the point c in sub-interval $(s, e]$
α	Learning rate for initial training of the NN
α'	Learning rate for the continual training of the NN
$\boldsymbol{\theta}$	Vector of neural network parameters
δ	Vehicle steering angle
$\dot{\psi}$	Vehicle yaw rate
$\hat{F}_i$	Vehicle friction tire force at wheel i
$\mathbf{u}$	Input vector (throttle, brake, steering) in a model-based observer
$\mathbf{x}_i$	Input feature vector in a time series data
$\mathcal{D}_{calib}$	Calibration set for conformal prediction

$\mathcal{D}_{test}$	Test set for conformal prediction
$\mathcal{D}_{train}$	Train set for conformal prediction
$\mathcal{Y}^*$	The selected change points (memory) based on the output time series set under study
$\mathcal{Y}$	The output time series set under study
μ	Tire-road friction coefficient
τ	Threshold set for detecting data distribution shift
ε	Noise term
$\tilde{g}$	The new calculated gradient after projection
a_x	Longitudinal acceleration
a_y	Lateral acceleration
d	Look-back window (history length)
d_{dim}	Dimension of the embedding layer in the Transformer model
d_{min}	Minimum distance between new data and memory points in input space
d_{multi}	Distance of the interval for keeping or removing multiple change points in each dimension in multi-output scenarios
g_k^*	The gradient of the loss function w.r.t the episodic memory
g_{k+1}	The gradient of the loss function w.r.t the current data
h	Number of hidden units in each layer in NN
l	Number of hidden layers in NN
m	Vehicle mass
m_i	Biased first moment in Adam optimizer
n	Number of observations
p	Number of input dimensions
q	Number of output dimensions
r	Yaw rate
$threshold$	The threshold value for multi-output memory selection
v_i	Biased second moment in Adam optimizer
$w_{fl}, w_{fr}, w_{rl}, w_{rr}$	Four wheel speeds of the vehicle
x_{norm}	Normalized input using min-max normalization
y_i	The target output in a time series data

Chapter 1

Introduction

1.1 Motivation

Recent advancements in statistical machine learning have significantly improved the accuracy and robustness of time series prediction across various domains, ranging from electric vehicle battery management and health life estimation to a robot’s model learning [1, 2]. However, achieving an accurate model fit using data-driven approaches becomes challenging when new environments containing new information are introduced frequently. In the context of time series analysis, vehicle state estimation is not an exception, as it enables precise control and enhanced safety in a variety of scenarios. In particular, the estimation of longitudinal and lateral velocities is essential for controlling the movement of a vehicle when sensor measurements are not reliable, which can occur due to adverse environmental conditions or faulty sensors. A major challenge arises from the intrinsic characteristics of the vehicle dataset, where new maneuvers in new data domains (performed in different conditions) are continuously added to the model in a sequential manner, necessitating efficient training of the model on new data or vehicle maneuvers while retaining previously learned information [3].

In kernel-based estimators such as Local Gaussian Process Regressions (L-GPRs), data are locally fitted, and predictions are performed based on the nearest local model to the input space [4]. However, these data-driven approaches require a large amount of data for accurate fitting, which is challenging in the vehicle data context. Also, these kernel-based approaches are regression models and do not leverage the knowledge of previous states (history of the fit) to predict current and future states. Moreover, their performance may be affected when one or more of the input states are faulty, resulting in an unreliable

estimation. Among data-driven estimators, Neural Networks have shown excellent performance in time series and data stream predictions, in general, and vehicle state estimation, in particular. NNs are capable of deriving nonlinear relationships between vehicle inputs, outputs, and past states. Continual learning in neural networks refers to the ability of a model to learn from a sequence of data sets or tasks in a continuous and incremental manner, without revisiting old data [5]. However, there is one significant challenge when it comes to the use of neural networks in continual learning scenarios; Neural Networks tend to forget the previously trained parameters (information) upon the addition of new data or batch of data, a phenomenon called catastrophic forgetting (CF) [6]. It is impossible to retrain the model with all previous and newly collected data due to time complexity. Therefore, a strategy is needed to efficiently train the model on new data, while preserving the previously learned information. The performance of existing continual learning methods are mostly limited to classification problems and fail to efficiently address the issue of catastrophic forgetting for regression problems. More specifically, there are limited work in the literature to address the problem of catastrophic forgetting in neural networks for time series regression tasks in general, and state estimation, in particular.

1.2 Objectives

In this work, a novel approach for time series prediction tasks is introduced to continually train the model on new data using neural networks, while preserving the previous information. We aim to address the detrimental impact of catastrophic forgetting in the context of time series and data stream regression scenarios, focusing on the applications on vehicle state estimation with large dataset. The proposed work combines statistical learning methods with memory- and regularization-based continual learning ideas to retain past knowledge. To achieve this, certain tasks are performed, including representative (coreset) selection, proper optimization method (minimizing a loss function), accurate selection of network structure and hyperparameters, and input or feature selection. There are certain steps to take to address this:

The general procedure consists of two major parts. First, a proper representative (coreset) selection approach is introduced to sample old or historical data by identifying representatives of the previous dataset based on the behaviour of the output function. Thus, the selected points are the most relevant points of the old dataset which the model has been previously trained on. The selection process should be computationally inexpensive and statistically accurate. In this regard, statistical methods for function approximation and change point detection in time series dataset are used to effectively find the most

relevant point as memory. Next, the new data and the selected memory points are added to the model jointly using an appropriate optimization algorithm. The goal is to ensure that the new information (data) is effectively integrated with the prior knowledge. The optimization process is based on memory-based continual learning frameworks, with improvements proposed in this work. The neural network structure and hyperparameters are then tuned to achieve optimal performance based on the initial dataset. Finding the optimal configuration for the neural network is done by several experiments and analysis of the intended application requirements. In this study, different advanced neural networks including Long-short-term memory (LSTM) and Transformer models are used to validate the proposed method’s performance. To select the optimal feature set for the neural network, both model-based and machine learning (ML)-based feature selection methods are used. In the model-based approach, the dynamic relationships between inputs and outputs are analyzed, while the ML-based approach uses input sensitivity analysis to identify the most important features. The main contributions of this thesis are summarized as follows:

- A novel continual learning framework is proposed, applicable to both univariate and multivariate output scenarios (Chapters 3 and 4). Ideas in memory-based knowledge retention and continual learning are used. This approach is relatively unexplored in the existing literature.
- Identifying a representative subset of historical data using statistical learning tools (Chapters 3 and 4). To the best of our knowledge, this technique has not yet been explored in the continual learning literature and constitutes a significant contribution to the field.
- An enhanced optimization algorithm is developed to minimize catastrophic forgetting while maximizing the learning of new tasks (Chapter 4).
- A comprehensive feature selection strategy is introduced, based on the sensitivity analysis of the input-output pair. In addition, a thorough uncertainty quantification of the neural network model is conducted using conformal prediction (Chapter 4).
- An adaptive state estimation framework is developed, capable of detecting distribution shifts in real time and dynamically switching between model-based and data-driven estimators (Chapter 5). The shift detection mechanism is done via representative selection strategy, introduced in the proposed continual learning framework.
- The proposed continual learning framework is applied to the problem of vehicle state estimation (Chapters 3, 4, and 5). Unlike prior works that primarily rely on traditional data-driven or local modeling approaches, this thesis leverages neural networks

to enable robust state estimation in lifelong learning settings, where the full dataset may not be available at once. Several advanced neural network architectures are applied to evaluate and validate the performance of the proposed method.

Figure 1.1 illustrates the overall structure of the proposed method and highlights the four main tasks to be accomplished. The first step involves selecting representatives from the historical or old dataset, ensuring that these selected points are the most informative and representative samples. This representative selection process leverages statistical analysis techniques in both univariate and multivariate time series scenarios. In the second step, upon the arrival of a new dataset containing additional information, a specific optimization algorithm is applied to jointly integrate the new data with the memory data, enabling efficient updating of the neural network parameters. Step three outlines the configuration and training procedures for the neural network, including the selection of appropriate hyperparameters. Finally, in the fourth step, the trained model is evaluated on both the new and old datasets to assess its adaptation to new data and to quantify the extent of catastrophic forgetting with respect to the old data.

1.3 Thesis Outline

In Chapter 2, a comprehensive review of the existing literature is presented. To start, previous work on vehicle state estimation using data-driven estimators, such as locally fitted methods is presented. Also, a review of continual learning approaches in classification problems is provided, since the proposed work is inspired by this field to develop a continual learning method in vehicle state estimation. Additionally, recent state-of-the-art approaches for vehicle state estimation tasks are introduced and discussed.

The proposed continual learning approach for univariate time series, termed Representative Selection for Continual learning in Time series tasks (ReSeleCT), is introduced in Chapter 3. First, the Average Gradient of Episodic Memory (A-GEM) method [7] is described and its limitations within the context of the intended application are discussed. Next, the proposed optimal coresct selection strategy is presented, which leverages statistical change point detection techniques. The subsequent section provides a detailed overview of the specific recurrent neural network architecture used. For the state estimation task, an LSTM neural network is adopted in this chapter. Finally, a comprehensive experimental analysis on vehicle state estimation is conducted, validating the effectiveness of the proposed method and comparing its performance against other state-of-the-art continual

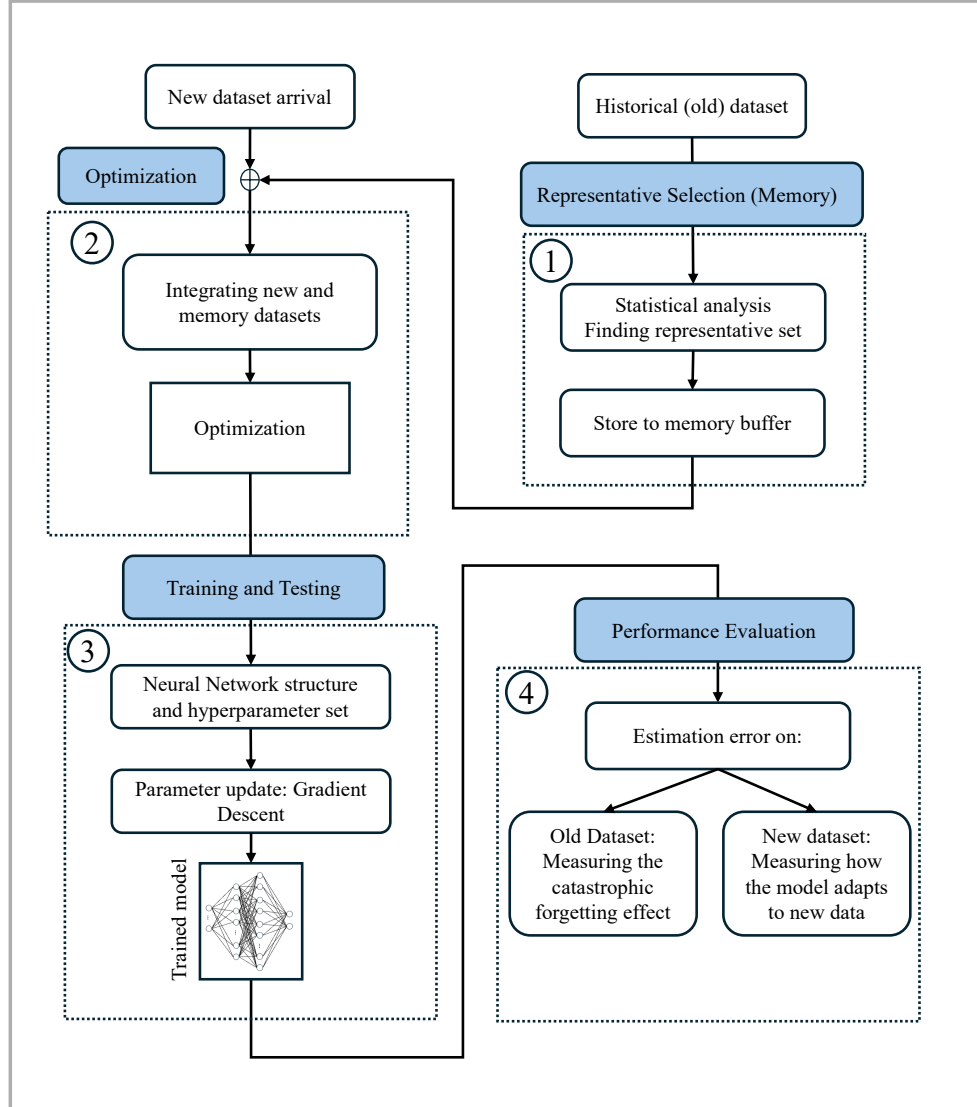


Figure 1.1: General scheme of the proposed method.

learning frameworks. The performance of the methods are evaluated under multiple addition of maneuvers, containing new information that needs to be added to the existing models.

In Chapter 4, the multivariate extension of the proposed method, termed EM-ReSeleCT, is introduced. Additionally, a refined version of the A-GEM algorithm, designed specifically for time series tasks, is presented. The chapter also includes comprehensive experimental validation of the proposed approach. For the neural network model, in this chapter a Transformer neural network model is used. Furthermore, uncertainty quantification is performed using conformal prediction to assess the reliability of the model's predictions. Additionally, a comprehensive sensitivity analysis on different vehicle signals is conducted to find the best set of features for the neural network inputs. Chapter 5 presents a practical implementation of continual learning using an adaptive state estimation framework, where task boundaries are identified based on the previously selected memory points. Extensive experimental analysis is performed to validate the detection of distribution shifts in the dataset. Additionally, this chapter provides a comprehensive evaluation of various neural network models for the state estimation task, comparing their performance in detail. Chapter 6, concludes the proposed work with a discussion on the results, and potential future work is explained.

Chapter 2

Literature Review

2.1 Introduction

Machine learning models have achieved remarkable success in recent years. Notably, deep learning models have shown exceptional capabilities across various machine learning domains, including natural language processing (NLP) and time series forecasting. In the field of time series analysis, numerous studies have explored the application of machine learning techniques for time series forecasting and prediction [8, 9, 10]. Generally, the application of machine learning models can be categorized into two distinct scenarios: stationary and non-stationary environments. In stationary environments, all training data is available in advance, and there is no distribution shift over time. In contrast, non-stationary environments are dynamic and subject to changes, necessitating the continuous adaptation or re-training of models to account for shifting data distributions.

Traditionally, machine learning models are trained on a specific dataset, referred to as the training set, and evaluated on a separate testing set. In the context of time series prediction and forecasting, numerous approaches have been explored in the literature, including support vector machines (SVMs), recurrent neural networks (RNNs), and autoregressive moving average (ARIMA) [11, 12, 13]. In such scenarios, the entire training dataset is typically available in advance, referred to as stationary environment, and there is no distribution shift between the training and testing sets, ensuring that the model operates under consistent data conditions. Traditional models often struggle to adapt to new datasets received under varying conditions, which may contain novel information. This challenge is particularly relevant in lifelong learning scenarios, where the model must continuously update its knowledge. The fixed structure of these traditional models limits

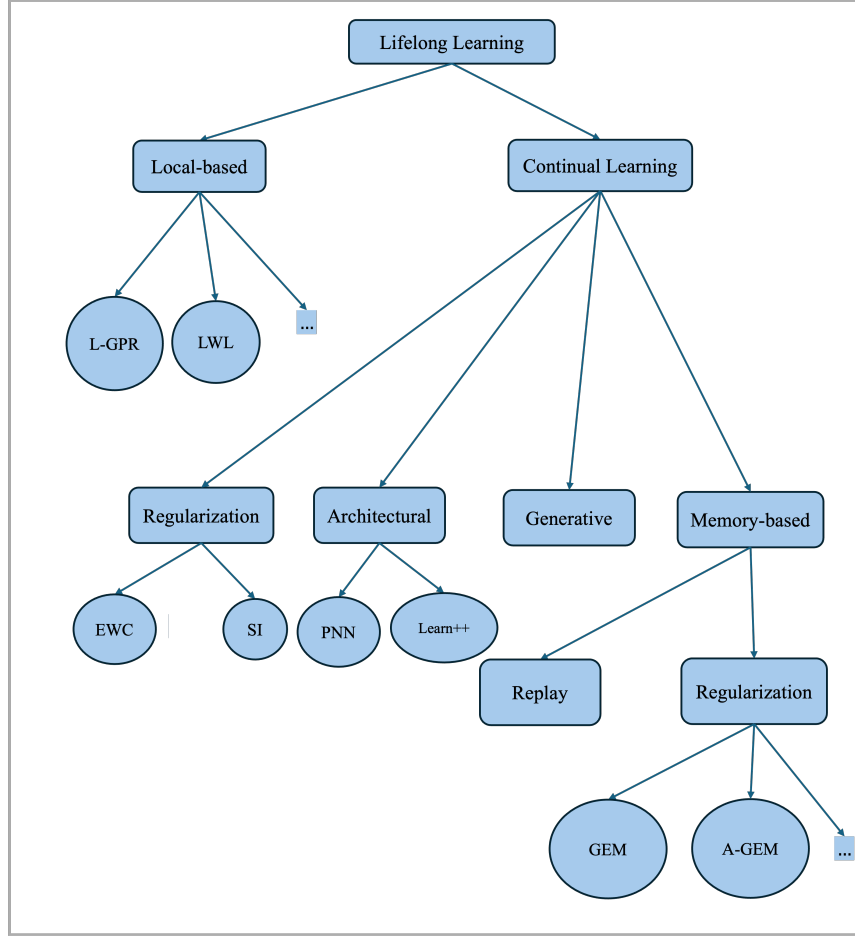


Figure 2.1: General categorizes of existing methods in lifelong learning scenarios.

their ability to handle such dynamic environments [14]. To overcome this limitation, various methods have been proposed in the literature to address lifelong learning scenarios. These approaches are primarily divided into two categories: local regression methods and continual learning techniques in neural networks. Figure 2.1 presents a flowchart illustrating the categorization of various methods in lifelong and continual learning scenarios, along with representative works from the existing literature.

2.2 Local Regression Methods

In local-based approaches, the idea is to fit a separate regression model on different data batches. The model can be linear, kernel-based or other regression techniques. By doing so, a new model is fitted in the new data space, and the previously learned models are remained unchanged. This is in contrast to traditional techniques, in which a single global model is fitted for the entire dataset. One such approach is the locally weighted learning (LWL) method [15]. LWL is capable of adapting to dynamical environments with non-stationary data, using different regression techniques such as locally weighted regression (LWR). The algorithm allocates local fits, allowing to addition of new data by new local models. Perhaps, the most reliable modification of LWL in incremental learning scenarios is locally weighted projection regression (LWPR) [16, 15, 17]. In LWPR, each input dimension is treated separately, and the regression is performed independently for each input dimension. This means that instead of performing regression in the original input space, LWPR projects the data into a lower-dimensional subspace before performing the regression. Based on the proposed model, weighting kernels are adjusted using local information to avoid interfering with new data fit to the previously fitted data. LWPRs were applied to several regression problems in sequential learning tasks from online robot dynamics learning to impedance control of robotic manipulators [18, 19]. They have been recently applied to vehicle state estimation [20].

Local Gaussian process regression (L-GPR) is another machine learning approach applied in robot dynamics estimation and online model learning control [21]. L-GPR uses the idea of locally fitting GPR on small data. By doing so, the challenge of high computational cost in GPRs on large datasets is avoided, and for newly collected data, a local model is defined with independent GPR parameters. Predictions are evaluated by finding the nearest local model to the data distribution. Local GPRs have been applied to several sequential learning domains including state estimation and safety-critical applications [4, 22, 23]. There are several critical problems in using local-based methods in data stream tasks and lifelong learning scenarios, in general, and vehicle state estimation, in particular. First, a large number of local models are often required to attain reliable estimations on test data, resulting in a high computational cost in prediction time. Second, these methods are unable to capture complex nonlinear relationships between the inputs, past and current output states. Also, the performance of the models can be negatively affected by faulty sensor measurements.

2.3 Continual Learning Methods

Upon training a model on new data or tasks, a pretrained neural network model fails to retain old information. A traditional approach aimed at addressing the forgetting problem in NNs was resource allocating networks (RAN) [24]. RAN starts with a small number of units (small network size), and dynamically allocates neurons to its structure. These networks are limited by their shallow networks, and their performance degrades, as the model complexity increases [25]. Another method was introduced in [26] based on measuring the importance of a neuron in radial basis neural networks, referred to as the growing and pruning radial bases function (GAP-RBF). The algorithm begins with zero hidden neurons, and adaptively adds and prune neurons, while new data is added. Among other methods, online sequential extreme learning machines (OS-ELM) and their variants [27, 28] are of great importance. However, these single hidden layer feedforward networks (SLFNs) cannot produce satisfactory performance in large and complex scenarios. Recently, several strategies have been proposed to address the issue of catastrophic forgetting in multilayer and deep neural networks. These methods are generally categorized into four main groups: Regularization-based, Architectural-based, Generative Replay, and Memory-based methods.

2.3.1 Regularization-based Methods

Regularization-based methods have become a popular method to reduce the effect of catastrophic forgetting and to enhance the model performance in continual learning scenarios. One common approach is elastic weight consolidation (EWC) [5]. In EWC, a penalization term was introduced in the cost function to avoid changes in important parameters of the neural network, where weight importance was measured by the Fisher Information (FI) matrix. In [29] a similar technique was used to avoid catastrophic forgetting by measuring parameter sensitivity called synaptic intelligence (SI), where the importance measure was based on the sensitivity of each weight with respect to the output function. EWC, SI, and their improved versions were recently applied to several real-world applications including image recognitions and classifications [30, 31].

Despite their excellent performance, the aforementioned methods are primarily applicable to image classification benchmarks and their efficiency tends to decrease in regression tasks. In [32], an adaptive continual learning technique in regression tasks named CLearR was introduced. After defining novelty and familiarity buffers captured from past and new data, the update process involves incorporating new data and subsequently evaluating the

model’s performance on the familiarity buffer to determine if its predictions remain accurate. If the model’s predictions on the familiarity buffer are unsatisfactory, an adaptive weight regularization technique based on EWC is employed to update the model using the new data. This method exhibits several limitations in regression problems: First, the regularization imposed by EWC tends to preserve the information of previously learned tasks at the expense of adding new information. While this is desirable to prevent catastrophic forgetting, it can lead to a rigid network that struggles to incorporate new patterns and variations in the data. As a result, the network’s flexibility may diminish after a certain number of tasks are introduced, leading to reduced performance. Second, the familiarity buffer (memory) is drawn based on recently arrived data, which is not always an optimal choice to find the representative of the old dataset. A better technique is required in this case. Additionally, it should be noted that the method was applied to a simplified case with only two input features as their final features, which may impact its generalizability to more complex datasets.

2.3.2 Architectural-based Methods

Another approach is to use a dynamic neural network structure during the learning of new tasks. In [33], a progressive neural network (PNN) was introduced, where an initial network was trained on initial tasks, and progressive networks were added based on new tasks arrival. Recently, a modified PNN was proposed, using the idea of reinforced continual learning and Bayesian continual learning to adaptively expand the neural network architecture [34]. A continual-learning framework that combines progressive neural networks with temporal convolutional networks to perform online state-of-charge (SOC) estimation across multiple lithium-ion battery chemistries and formats [35]. Unfortunately, using PNNs and their variants in regression tasks, particularly for the intended application, results in a large network size after adding a few specific tasks. Another architectural-based approach was introduced in [36], known as Learn++. This ensemble learning method comprises multiple weak learners, each corresponding to a specific new task. The overall prediction is obtained through a weighted average of the outputs from these individual classifiers. A recent extension of Learn++ incorporates recurrent neural network structures [37], which has also been applied in various industrial domains [38]. However, these methods are inherently tied to specific model architectures, which may limit their generalizability. Furthermore, when faced with a large number of new tasks, the proliferation of weak learners can lead to significant computational overhead.

2.3.3 Generative Replay

Storing past data as a memory is not always practical due to security concerns or storage limitations. One solution is to use generative models to generate past data during the learning on new data without storing memory [39]. Generative models focus on identifying the underlying data distribution in the feature space, thereby generating new (fake) samples in the input data distribution [40]. Therefore, the task is to keep the probability distribution of the fake generated samples by the model similar to the probability distribution of the real data. In [39], a dual model architecture, consisting of a generative model called “generator” and a solving model called “solver,” was proposed, with the aim of sampling previous data to avoid catastrophic forgetting. Generative replays were also applied to time series classification benchmarks, such as human activity recognition tasks [41, 42]. A deep generative replay-based continual learning framework was proposed in [43] for class-incremental surface electromyogram (sEMG) pattern recognition, aiming to address catastrophic forgetting when adding new classes without retraining on historical data. However, there are two main issues, when applying these models in the intended application: First, generative models generate data depending on the distribution of the inputs, but there is no specific method to select informative data from old tasks, as some inputs may be more informative than others. Therefore, the generated inputs may exhibit similarity or redundancy. Second, a separate model is needed for every batch of data (a distinct vehicle maneuver in this work), leading to a large number of models after multiple tasks.

2.3.4 Memory-based Methods

A more effective strategy to mitigate catastrophic forgetting is to store data from past information and use them in the new learning task. These stored data can be directly used along with the new data to train the model, a technique called memory replay, or they can be used as regularization terms in the optimization [44].

Memory replay: Memory replay involves retraining the model on stored data in memory during the training on the new task. There are different strategies for storing data in memory. One common approach is experience replay, where memory points are selected by random sampling [45]. However, random sampling may not always accurately represent the old data. An alternative approach to find representatives of previously learned knowledge was proposed based on capturing the coreset of the old data by selecting a weighted subset via bilevel optimization [46]. In [47], the importance of the samples was defined based on their diversity in the current (new) task. It is worth noting that, all of the aforementioned

methods were applied to classification benchmarks and may not be effective in regression tasks, where the behavior of the output function is of greater importance. Therefore, a more reliable memory selection strategy is needed for regression scenarios.

Optimizing via memory regularization: The risk of overfitting on the stored data, when replaying in the learning is a potential concern. To address this issue, a memory regularization-based approach called the gradient of episodic memory (GEM) has been proposed [48]. In GEM, after selecting the desired memory, the idea was to minimize the loss on new data using stochastic gradient descent (SGD), while maintaining alignment with the gradient of the memory. To do so, an inequality constraint was defined to calculate the inner product between the gradient vector of the new data and the episodic memory in each training epoch. A positive inner product indicates that the two gradients point in the same direction in the parameter space, and the new gradient is used to update network parameters using the SGD. However, if the inner product is negative, using the new gradient results in forgetting. In this case, the new gradient is projected onto the nearest gradient vector that points in the same direction as the gradient of the episodic memory. To find the projected gradient, GEM uses dynamic programming, which is computationally expensive. An alternative approach known as the average gradient of episodic Memory (A-GEM), and its improved scheme were proposed, aiming to find the projected gradient in a more efficient manner [7, 49]. GEM and A-GEM have been applied to some applications [50, 51].

2.3.5 Continual Learning Applications to Time Series Forecasting

Recent advances in continual learning in neural networks have extended to time series tasks [52]. In [53], the authors presented an improved version of CleaR named V-CleaR to address continual learning with focus on power forecast. A new framework called Fast and Slow learning Networks (FSNet) for online time series forecasting was proposed in [54], where a per-layer adapter that leverages the exponential moving average (EMA) of layer gradients and an associative memory that stores and retrieves adaptation coefficients corresponding to recurring patterns was presented. However, the approach was limited to the specific structure (in this case, a Temporal Convolutional Network), so its performance might be sensitive to the choice of the neural network architecture. Also, the scenario considered in the paper was online learning, where the learning is carried out in a series of rounds. In each round, the model receives a look-back window and is then required to predict the forecast window. Once the actual results are revealed, they are used to refine the model’s predictions for future rounds, which is different from task specific continual learning scenario, where

each task is arrived and the model is trained and predicts independently. In [55], a continual learning framework with graph neural networks (GNN) was proposed to estimate the traffic flow which is particularly designed for the mentioned application and cannot be generalized to different applications in continual learning for time series forecasting. Additionally, a continual learning augmentation (CLA) that accumulates models of past states was introduced in [56], which was specifically designed for financial market forecasting to drive long-term financial decisions.

Most existing continual learning methods for time-series forecasting are limited to their target or intended applications and underlying architectures. For example, V-Clear [53] and FSNet [54] depend on specific backbones (power-forecasting networks or Temporal Convolutional Networks), making them vulnerable to architectural choices and difficult to transfer across domains. Similarly, the online-learning formulation in FSNet, where the model updates only after each look-back/forecast cycle, differs fundamentally from task-based continual learning, where tasks arrive independently and must be learned accordingly.

2.4 A Review of Advanced Machine Learning Methods in Vehicle State Estimation

In the context of vehicle state estimation, numerous approaches have been explored in the literature, including both purely data-driven machine learning methods and hybrid techniques that combine data-driven and model-based strategies. Gaussian process regression is a widely used probabilistic, non-parametric regression technique that has found extensive application in state estimation, particularly for vehicles [57, 58, 59]. In [59], the authors used GPR for lateral force estimation in intelligent tire systems. In [58] a hybrid approach was proposed that integrates a model-based observer with GPR to improve vehicle state estimation under both normal and emergency conditions. Additionally, in [60], the authors combined a feedforward neural network with a dynamic-based method using recursive least squares (RLS) for vehicle mass estimation, where the outputs of both methods were fused using fuzzy neural networks.

Neural networks have been widely applied to vehicle state estimation problems. In [2] a deep recurrent neural network for brake pressure estimation in electric vehicles was proposed. More recently, Transformer-based neural networks have attracted significant attention due to their effectiveness in time series forecasting in terms of both accuracy and computational efficiency. In [61] a Transformer architecture for state of charge (SoC)

estimation in electric vehicle batteries was introduced, adapting the encoder-decoder structure originally proposed in [62] to the estimation context. In addition, several studies have explored physics-informed neural networks [63, 64], which combines mathematical models into the neural network framework—either through the loss function or other integration methods, to improve estimation accuracy.

2.5 Summary

In this Chapter, a comprehensive review of the literature on different lifelong and continual learning methods, to continuously training the model on new information, while preventing catastrophic forgetting was presented. Additionally, the application of continual learning methods in time series methods scenarios were presented. Despite the significant attention to the continual learning methods in recent years, there is limited studies on continual learning in time series regression tasks such as state estimation, forecasting, etc. Based on a comprehensive review of the existing methods in continual learning, developing a suitable approach for the specific application is a need.

Chapter 3

Representative Selection and Continual Learning in Time Series Regression Tasks

3.1 Introduction

In this chapter, the proposed method for continual learning on time series datasets in univariate output space is outlined. First, a memory-based continual learning optimization algorithm called average gradient of episodic memory (A-GEM) is used as a tool to enable effective continual learning of new data while mitigating catastrophic forgetting. Next, a representative selection strategy is introduced to efficiently select memory points from historical time series data. Finally, experimental analyses and validations are conducted on the task of vehicle state estimation, with the proposed method compared against other state-of-the-art continual learning algorithms.

As previously discussed in Chapters 1 and 2, a comprehensive review of current methods on continual learning suggested that memory-based methods are likely to produce the most favorable performance and are best suited for the intended application. This hypothesis is supported by several key factors: First, regularization-based methods limit the model's flexibility to learn new tasks, after a certain number of data batches introduced to the network. Additionally, regularization parameters play an important role in the model's performance. Second, architectural-based methods can present a significant problem with network size as the training progresses. As shown in Figure 3.1, a widely used approach

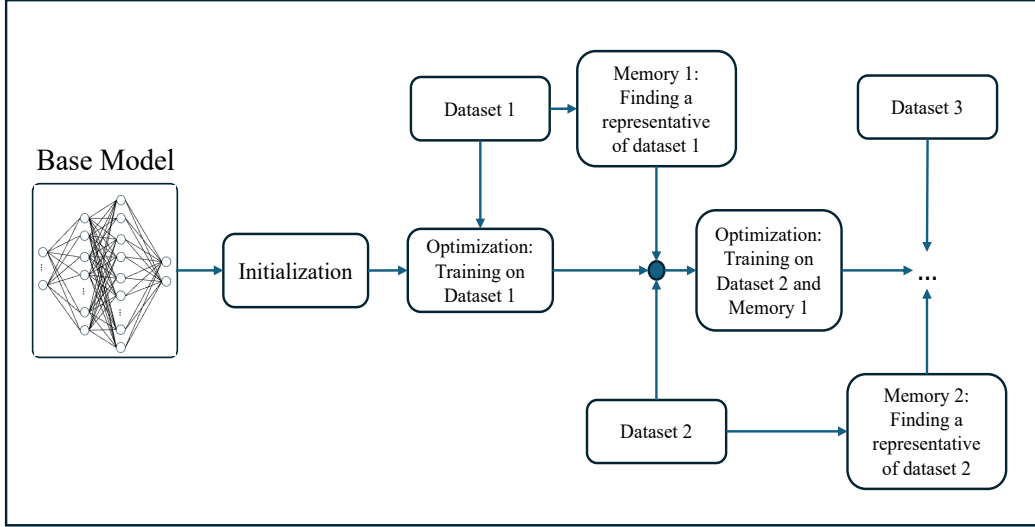


Figure 3.1: The scheme of a memory-based continual learning framework.

for memory-based continual learning consists of two main stages: (1) finding the representatives of the old task, and (2) optimization. The step-by-step process is elaborated below:

1. **Initialization:** A base model of the neural network is defined (i.e., setting hyperparameters including the number of layers and number of nodes), and the parameters are initialized.
2. **Initial training:** The first dataset (normally a large dataset) is added to the neural network, and network parameters are updated subsequently.
3. **Representative (Coreset) selection:** An appropriate method is used to select representatives from the old dataset. The stored representatives are the key information of the historical data.
4. **Optimization:** Once a new task is introduced to the model, the parameters are updated so that the prior information is retained while integrating the new task, and avoiding heavy computations. This step uses both the stored memory points and the new task to optimize a predefined cost function.
5. **Iteration:** The entire process is repeated once the network identifies a new task.

By following the above process, catastrophic forgetting is mitigated by preserving the information of the previously trained model while introducing new tasks to the neural network. This approach effectively enables continual learning in task-incremental data stream scenarios. In the following sections, the continual learning problem in the univariate case is formulated, and an effective optimization technique (A-GEM) is discussed. Next, the proposed framework for efficiently finding representatives for this optimization approach is introduced. Note that throughout this chapter, the proposed approach is referred to as **ReSeleCT (Representative Selection for Continual learning in Time-series scenarios)**. Parts of this chapter are based on our published work in [65].

3.2 Formulating Continual Learning in Time Series Regression

Consider a stream of observations denoted by $(\mathbf{x}_i, y_i)$, for $i = 1, 2, \dots$, where $y_i \in \mathbb{R}$ represents the target output and $\mathbf{x}_i \in \mathbb{R}^p$ denotes the corresponding input feature vector. For all $i = d + 1, d + 2, \dots$ we assume the output follows:

$$y_i = f(\mathbf{x}_i, H_i(D)) + \epsilon_i, \quad (3.1)$$

where D is an available dataset, f is an unknown function to be estimated, ϵ_i is the additive noise of observation i , d is a given integer, which indicates the degree of temporal dependencies among $(\mathbf{x}_i, y_i)$ s, and

$$H_i(D) = \{(\mathbf{x}_j, y_j) \in D; i - d \leq j < i\}.$$

The objective is to estimate the function f using the available dataset D . As previously discussed in Chapter 2, among different potential methods in the literature, neural networks have proven to be highly effective for estimating f in time series applications. By adopting a neural network framework, we implicitly assume that f in (3.1) can be accurately approximated by a neural network parameterized by $\boldsymbol{\theta}$. Accordingly, the function is represented as $f(\cdot; \boldsymbol{\theta})$ instead of f and the task of estimating f is reduced to estimating the parameter vector $\boldsymbol{\theta}$. We define the squared error loss for the i -th observation as $\ell_i(\boldsymbol{\theta}) = (y_i - f(\mathbf{x}_i; \boldsymbol{\theta}))^2$. To estimate $\boldsymbol{\theta}$, typically the empirical risk function is minimized:

$$R(\boldsymbol{\theta}, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \ell_i(\boldsymbol{\theta}), \quad (3.2)$$

where $|D|$ denotes the total number of data points in the dataset D . Suppose a sequence of k datasets is observed from a particular experiment, where each dataset contains time series data corresponding to a distinct data domain. For notational convenience, let

$$\begin{cases} D_1 = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_1}, y_{n_1})\} \\ D_2 = \{(\mathbf{x}_{n_1+1}, y_{n_1+1}), \dots, (\mathbf{x}_{n_2}, y_{n_2})\} \\ \vdots \\ D_k = \{(\mathbf{x}_{n_{k-1}+1}, y_{n_{k-1}+1}), \dots, (\mathbf{x}_{n_k}, y_{n_k})\} \end{cases} \quad (3.3)$$

If all datasets are collectively available in advance, i.e. $D = D_1 \cup D_2 \cup \dots \cup D_k$, then a neural network can be trained using the complete set of $n = \sum_{i=1}^k n_i$ data points in D . However, in a continual learning setting, the datasets $D_1, D_2, \dots, D_k, \dots$ arrive sequentially over time and are not accessible all at once. As new data becomes available, the model must be updated accordingly. In practical application domains, such as in vehicle state estimation, this ongoing model adaptation can be computationally demanding and even impractical, particularly due to memory limitations that prevent storage of all incoming data batches. Let D_1 represent the first batch of data received, and let $\hat{f}_1 = f(\cdot, \hat{\theta}_1)$ define the neural network estimate of the function f based solely on D_1 , where the parameters are obtained by minimizing the empirical risk, i.e. $\hat{\theta}_1 = \underset{\theta}{\operatorname{argmin}} R(\theta, D_1)$. The core concept of memory-based continual learning [48] is to identify m_1 landmarks from dataset D_1 , $m_1 \ll n_1$, such that this subset effectively captures the essential information in D_1 for estimating f . Denote the selected landmarks as:

$$D_1^* = \{(\mathbf{x}_1^*, y_1^*), \dots, (\mathbf{x}_{m_1}^*, y_{m_1}^*)\}. \quad (3.4)$$

These selected landmarks in D_1^* are then used in conjunction with the next incoming dataset D_2 , to provide an updated estimate $\hat{f}_2 = f(\cdot, \hat{\theta}_2)$ while limiting the forgetting of the previously learned function $\hat{f}_1$. Therefore, the new parameter estimate $\hat{\theta}_2$ is obtained by solving the following constrained optimization problem:

$$\hat{\theta}_2 := \underset{\theta}{\operatorname{argmin}} R(\theta, D_2) \quad \text{s.t.} \quad R(\theta, D_1^*) \leq R(\hat{\theta}_1, D_1^*).$$

3.3 Optimization with Average Gradient of Episodic Memory

After receiving the t -th dataset D_t , a well-established method for minimizing empirical risk while preserving the performance on previously learned data is the average gradient

episodic memory (A-GEM) algorithm [7]. This approach calculates the parameter $\hat{\boldsymbol{\theta}}_t$ by solving the following constrained optimization problem:

$$\hat{\boldsymbol{\theta}}_t := \arg \min_{\boldsymbol{\theta}} R(\boldsymbol{\theta}, D_t) \text{ s.t. } R(\boldsymbol{\theta}, \bigcup_{i=1}^{t-1} D_i^*) \leq R(\hat{\boldsymbol{\theta}}_{t-1}, \bigcup_{i=1}^{t-1} D_i^*), \quad (3.5)$$

resulting in the updated model estimate $\hat{f}_t = f(\cdot, \hat{\boldsymbol{\theta}}_t)$. To implement A-GEM, let

$$\begin{aligned} g_k^* &= \frac{\partial}{\partial \boldsymbol{\theta}} R(\boldsymbol{\theta}, \bigcup_{i=1}^k D_i^*)|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k} \\ g_{k+1} &= \frac{\partial}{\partial \boldsymbol{\theta}} R(\boldsymbol{\theta}, D_{k+1})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k}. \end{aligned}$$

During each training epoch, the process involves computing the inner product between the newly obtained gradient g_{k+1} and the episodic memory gradient g_k^* . Note that g_k^* is the average of the gradient of the episodic memory of the previous tasks. If the inner product $\langle g_k^*, g_{k+1} \rangle$ is negative, this indicates that the angle between the new and memory gradients is greater than 90 degrees. In such a case, the new gradient g_{k+1} is projected onto the closest gradient $\tilde{g}$ that satisfies $\langle \tilde{g}, g_k^* \rangle \geq 0$ [7]. This projection is performed to execute the following optimization:

$$\min_g \|g_{k+1} - g\|_2^2 \quad \text{s.t.} \quad \langle g, g_k^* \rangle \geq 0, \quad (3.6)$$

which has the closed form solution

$$\tilde{g} = g_{k+1} - \frac{\langle g_{k+1}, g_k^* \rangle}{\|g_k^*\|_2^2} g_k^* \quad (3.7)$$

Proposition: Using (3.7), the new calculated gradient and the memory gradient are orthogonal. That is,

$$\langle g_k^*, \tilde{g} \rangle = 0 \quad (3.8)$$

Proof:

$$\begin{aligned} \langle g_k^*, \tilde{g} \rangle &= \left\langle g_k^*, \left(g_{k+1} - \frac{\langle g_{k+1}, g_k^* \rangle}{\langle g_k^*, g_k^* \rangle} g_k^* \right) \right\rangle \\ &= \langle g_k^*, g_{k+1} \rangle - \frac{\langle g_{k+1}, g_k^* \rangle}{\langle g_k^*, g_k^* \rangle} \langle g_k^*, g_k^* \rangle \\ &= \langle g_k^*, g_{k+1} \rangle - \langle g_{k+1}, g_k^* \rangle = 0 \end{aligned} \quad (3.9)$$

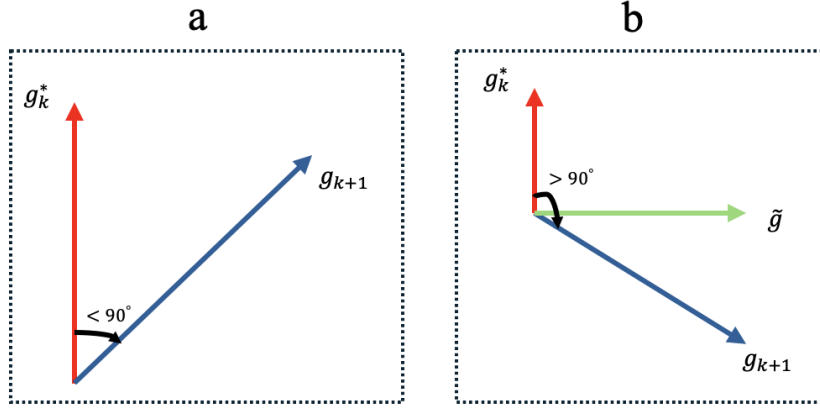


Figure 3.2: The position of the two gradients in parameters space; (a): The scenario where the inner product between the two gradients is positive. (b): The scenario where the inner product between the two gradients is negative, along with the new projected gradient.

Figure 3.2 shows the directions of the new gradient and the gradient of the memory for the two scenarios specified. Within the A-GEM framework, it is assumed that the stored memory samples accurately capture the behavior of the previous tasks. This gives rise to two fundamental questions that need to be considered:

- What defines the optimal subset of the historical data in time series datasets?
- How to identify and capture the best subset?

Originally, A-GEM used a random subset of previous or historical dataset as the episodic memory to retain past information. However, this approach may not be effective enough to select representatives of the historical data, as some points are more informative than others. Therefore, a more principled strategy is needed to select a coreset that best represents the most relevant historical data. Different methods for coreset selection have been explored in the literature, in semi-supervised learning [66], continual learning [46, 67], and active learning [68], most of which are primarily designed for classification tasks, where preserving the continuous structure of the output function is not a central concern since the target is not continuous. In contrast, for time series regression tasks, the focus is placed on maintaining historical knowledge by selecting representative samples based on the behavior of the output function rather than solely focusing on the inputs. The proposed methodology is particularly beneficial as it allows for preserving previous information predominantly derived from the output in regression problems.

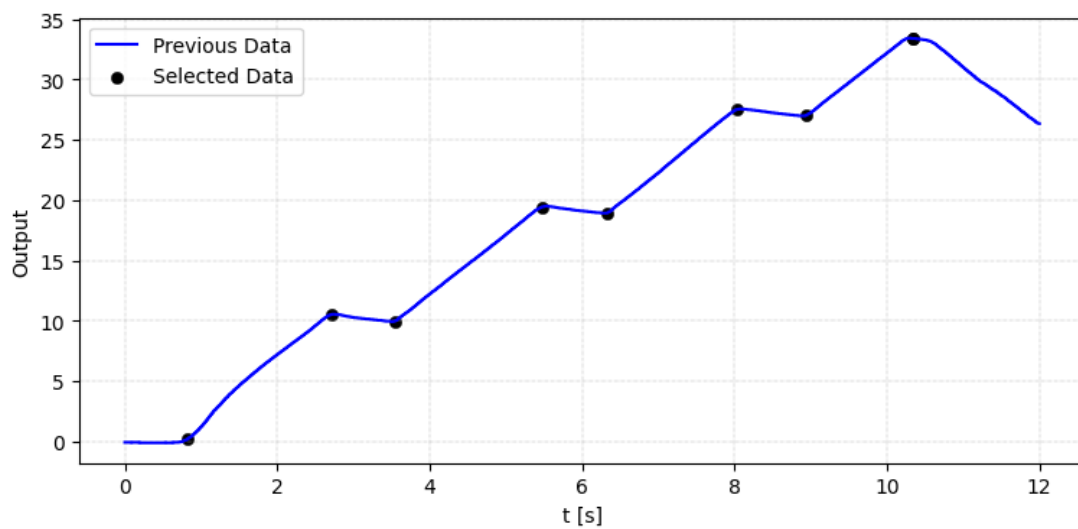
3.4 Capturing Key Information of a Function via Change Point Detection

For a given dataset $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)\}$, let $\mathcal{Y} = \{y_1, \dots, y_T\}$ denote its output dataset. We argue that the most informative samples in the dataset D generally correspond to points where the output pattern $\mathcal{Y}$ exhibits sudden changes. By differentiating between significant changes or shifts from noise-induced changes, the locations that contain meaningful information can be identified for accurately estimating the output function explaining the time series y_t . These identified points are also particularly prone to inducing catastrophic forgetting. Let $k \ll T$ be the number of detected change points in $\mathcal{Y}$, and $0 = c_0 < c_1 < \dots < c_k < c_{k+1} = T$ represent the location of these change points so that $\mathcal{Y}^* = \{y_{c_1}, \dots, y_{c_k}\}$ is the selected memory points (i.e., the change points). Refer to Figure 3.3(a) for a visual illustration. Suppose $y_t = \mu_t + \sigma_t \varepsilon_t$, for $t = 1, \dots, T$, where μ_t is a local linear function of the following form:

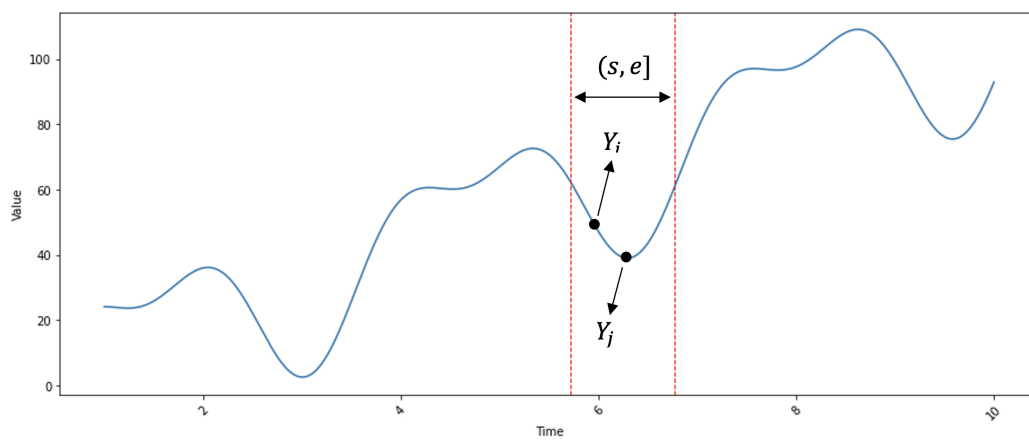
$$\mu_t = \begin{cases} m_1 t + b_1 & 1 \leq t < c_1 \\ m_2 t + b_2 & c_1 \leq t < c_2 \\ \vdots & \vdots \\ m_{k+1} t + b_{k+1} & c_k \leq t \leq T, \end{cases} \quad (3.10)$$

ε_t is the Gaussian noise, σ_t is the standard deviation of y_t at time t , and m_i and b_i are the parameters that define the local linear fit between the two consecutive change points $c_{i-1} < c_i$. The goal is to estimate a proper set of change points $(c_1, \dots, c_k)$ that best describes (3.10). By recording points $\mathcal{Y}^* = \{y_{c_1}, \dots, y_{c_k}\}$ as memory markers and eliminating the extensive data in between for future learning, a substantial amount of redundant information have been effectively eliminated while retaining essential historical information.

To detect changes in the mean function μ_t of y_t , in this study, the narrowest over a threshold (NOT) [69] method is used as a tool to find change points (coreset) $\mathcal{Y}^*$ of the old or historical dataset $\mathcal{Y}$. The algorithm starts by dividing the dataset $\mathcal{Y}$ into M randomly drawn sub-samples. This is performed over the indices of the observations, as illustrated in Figure 3.3(b), where each interval $(s, e]$ corresponds to the consecutive observations $y_{s+1}, \dots, y_e$. Assuming that at most one change point exists within each interval, the Gaussian generalized likelihood ratio (GLR) test statistic can be computed for all potential change point locations inside each interval $(s, e]$. If there is no change point in the interval, then the likelihood function $L(y_{s+1}, \dots, y_e; \theta)$ is maximized to estimate



(a)



(b)

Figure 3.3: (a): An example of desired locations to store in memory. (b): change point detection in each randomly drawn interval in a time series data.

the intercept and the slope parameters $\eta = (b, m)$ of the mean function μ_t for $t \in (s, e]$. If there is a change point at $c \in (s, e]$ the mean function μ_t will have different intercepts and slopes in the intervals $(s, c]$ and $(c, e]$ and the corresponding parameters are denoted with η_1 and η_2 . Consequently, the GLR test statistic for evaluating the hypothesis of no change point versus the presence of one change point at position

(c)

is given by

$$\mathfrak{R}_{(s,e]}^c(\mathcal{Y}) = 2 \log \left[\frac{\sup_{\eta_1, \eta_2} \{L(y_{s+1}, \dots, y_c; \eta_1) L(y_{c+1}, \dots, y_e; \eta_2)\}}{\sup_{\eta} L(y_{s+1}, \dots, y_e; \eta)} \right] \quad (3.11)$$

Where $\mathfrak{R}_{(s,e]}^c(\mathcal{Y})$ is the generalized likelihood ratio value of point c at sub-interval $(s, e]$ of dataset $\mathcal{Y}$. In Figure 3.3(b), between the two selected points y_i and y_j the following statement is expected:

$$\mathfrak{R}_{(s,e]}^i(\mathcal{Y}) \geq \mathfrak{R}_{(s,e]}^j(\mathcal{Y})$$

Applying the GLR test for all data in each interval $(s, e]$, NOT estimates a change point by finding the argument that maximizes Eq. (3.11):

$$\mathfrak{R}_{(s,e]}(Y) = \max_{c \in \{s+2, \dots, e-2\}} \mathfrak{R}_{(s,e]}^c(Y) \quad (3.12)$$

Therefore, the potential change point in each interval is the one that carries the highest likelihood ratio among all the other points in the interval. After finding all potential change points for all M randomly drawn intervals, only those candidates whose GLR values exceed a predefined threshold are retained and the rest are excluded from the process. From the retained candidates, the point located in the narrowest interval is selected. For more details see [69].

The proposed method (ReSeleCT), uses change point detection to capture the most informative and significant points within old or historical data. Here, a data point is considered “representative” based on two key aspects; First, the selected points should minimize the error of the linear fit of the model between two consecutive change points (estimating the function parametrically) while selecting the minimum number of points. This ensures that the representation is both accurate and efficient, facilitating the estimation of the output function (for the application in vehicle state estimation, estimating the

output function for each maneuver). As illustrated in Figure 3.3(a), the error of a linear fit between two consecutive change points is minimized. The NOT method estimates the function between the two pair of consecutive change points parametrically (with linear parameter model in this study). To ensure both minimum error and the selection of the minimum number of points is selected, the NOT method uses the strengthened Schwarz Information Criterion (sSIC) as a model selection method to find the best model. Second, the selected points should correspond to locations where significant changes in the behavior of the output occur. These changes are crucial as they reflect shifts in the underlying data distribution, essential for robust model updating and knowledge retention. Specifically, the Narrowest-Over-Threshold (NOT) method is used to identify these change points, using a Gaussian generalized likelihood ratio (GLR) test to detect points with the highest likelihood of representing true changes. By focusing on these criteria, it is ensured that the selected subsets retain the critical information necessary for accurate and efficient continual learning.

3.4.1 Representative Selection in Time Series Dataset

So far, the proposed approach has involved identifying informative points within a function. It is crucial to recognize that a singular point is insufficient as a memory point for future learning, particularly given the use of recurrent neural networks, where the time sequence holds significance due to the temporal dependencies among the data points. In this regard, the time sequences are selected as memory points in which the identified change point is located. Figure 3.4 shows the proper time sequence to capture as a memory point. This idea is guided by the primary objective of recurrent neural networks which is to uncover inherent patterns within a time sequence. The act of capturing and storing these particular time sequences that showcase alterations in their patterns significantly enhances the capacity to preserve crucial past information.

Next, the detail of how ReSeleCT algorithm is used to identify the optimal subset from historical data is outlined. These representatives are subsequently used to retrain the model alongside new data collected within a new data domain, using a specific optimization technique. As will be discussed in the experiments, a recurrent neural network model is trained on a large dataset consisting of different time series tasks. Each time series dataset consists of different sensor measurements as input features, along with a desired output for estimation. After training the model on the initial dataset, ReSeleCT identifies representatives from the historical data for retraining with the newly collected data. Algorithm 1 outlines the overall process of coreset selection for continual learning. In the initial phase, the algorithm begins with a dataset from domain D , which may include various

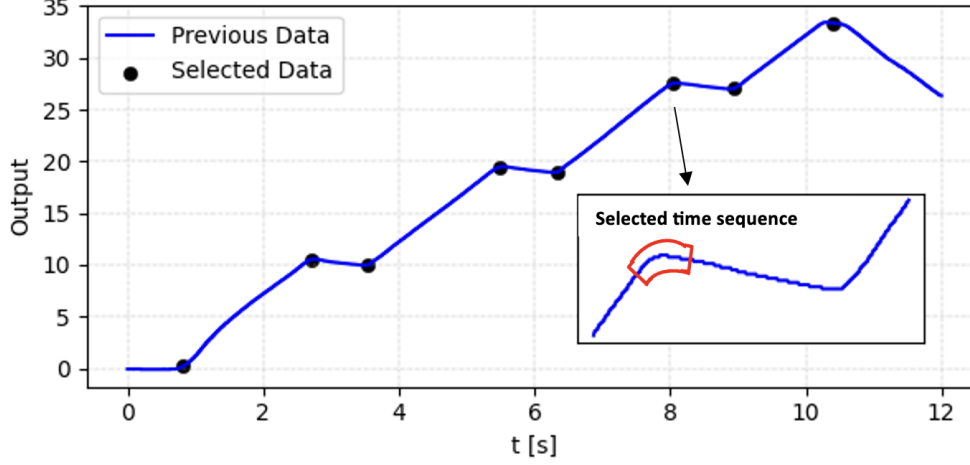


Figure 3.4: Proper time sequence to capture in memory.

sub-domains. Step 1 involves training the model on this initial dataset. Subsequently, for all time series datasets within domain D , steps 3 to 6 are performed, where representatives are selected and stored in memory. The selection of these representatives can either be limited to a specific number for each maneuver (buffer mode) or determined automatically using strengthened Schwarz information criterion (sSIC). Steps 8 and 9 include introducing the model to the new data domain and conducting training using the combined dataset (selected memory points and the new dataset) with A-GEM optimization. Finally, in step 10, the model repeats this process to capture representatives from the newly collected data and add them to memory. Figure 3.5 shows the step-by-step flowchart of the proposed method.

While the proposed approach primarily focuses on vehicle state estimation, the ReSeleCT methodology is inherently versatile and applicable to a broad range of time series regression problems. For instance, in electric vehicle (EV) battery management, ReSeleCT can be used to continually learn and predict the State of Health (SoH) and State of Charge (SoC) of batteries. By identifying and storing representative points in battery data streams, ReSeleCT ensures accurate and up-to-date predictions, even as new data is continuously introduced. Beyond time series data, ReSeleCT can be adapted to any regression problem involving data streams. The core principle of selecting representative subsets of data to mitigate catastrophic forgetting is not limited to any specific data type. For example, in financial forecasting to predict stock prices or market trends, the model

Algorithm 1: Continual Learning with ReSeleCT

Input: Initial time series datasets $D_1, \dots, D_k$, where D_i is the i^{th} time series dataset, consisting of sensor measurements from inputs and the desired output.

Output: Trained model, representatives of the data

- 1 Initial training of the model based on the initial dataset (D).
 - 2 **for** D_i *in* $\{D_1, D_2, \dots, D_k\}$ */* where D_i (Eq. 3.3 consists of sensor measurements X_i and Y_i (Eq. 3.1) */*
 - 3 **do**
 - 4 Apply change point detection to the sensor measurements from the desired output
 - 5 After finding several models (each model with a different number of captured points), select the best via sSIC
 - 6 $D^* \leftarrow D_i^*$: Store the captured points to the memory
 / $D_i^* = \{(x_{i1}^*, y_{i1}^*), (x_{im}^*, y_{im}^*)\}$ */*
 - 7 **Return** $D^* = \{D_1^*, D_2^*, \dots, D_k^*\}$
 - 8 Receiving a dataset in new domain (D_{k+1}).
 - 9 Train the model using the joint dataset ($D^* \cup D_{k+1}$) and the A-GEM optimization algorithm.
 - 10 Apply steps 4, 5, and 6 for the newly collected dataset.
 - 11 Wait for the next time series dataset.
-

can continually learn from data streams in different domains and learn new information while maintaining the previously learned knowledge using representative selection process in ReSeleCT. Additionally, the proposed method can be applied to data stream problems where outputs are not necessarily dependent on time but on other variables in the stream. For instance, in telecommunications, ReSeleCT can analyze data streams dependent on frequency rather than time. In this scenario, ReSeleCT continually learns new information to predict frequency-dependent outputs. By selecting representative points from frequency-dependent output streams, ReSeleCT is used for continual learning, thereby preventing catastrophic forgetting. This demonstrates the flexibility and wide applicability of the ReSeleCT approach beyond traditional time series regression tasks.

3.5 Evaluation and Testing

In this section, the performance of the proposed method, ReSeleCT, is evaluated in estimating the vehicle’s longitudinal and lateral velocities. The step-by-step procedure used to evaluate the method’s performance is illustrated in Figure 3.6. For the estimation of each state, there are two phases to consider. The first phase involves preprocessing and initial training of the model. The second phase corresponds to the continual learning scenario, where new data domains are introduced to the model and the evaluation of the model’s performance in estimating the output corresponding to the new maneuvers, and previously added data is evaluated.

3.5.1 Dataset Description

The dataset used in this experimental analysis was collected through experiments conducted on an electric Chevrolet Equinox shown in Figure 3.7. Several maneuvers were performed including driving in a circle, executing left and right turns, and driving up a hill. These maneuvers were carried out under different road conditions, including dry and wet surfaces. During each maneuver, data from the Inertial Measurement Unit (IMU), Global-Positioning-System (GPS), and wheel sensors were collected. The signal measurements were sampled at a frequency of 200 Hz for all maneuvers. Table 3.1 provides a detailed description of the dataset used, including the number of maneuvers selected for training and testing, and the total size of the collected data. As shown in the table, two maneuvers were carefully selected as the test sets for the estimation of both longitudinal (V_x) and lateral (V_y) velocities. These maneuvers were specifically selected to be similar to the ones used during the initial training phase. For estimation of both longitudinal and

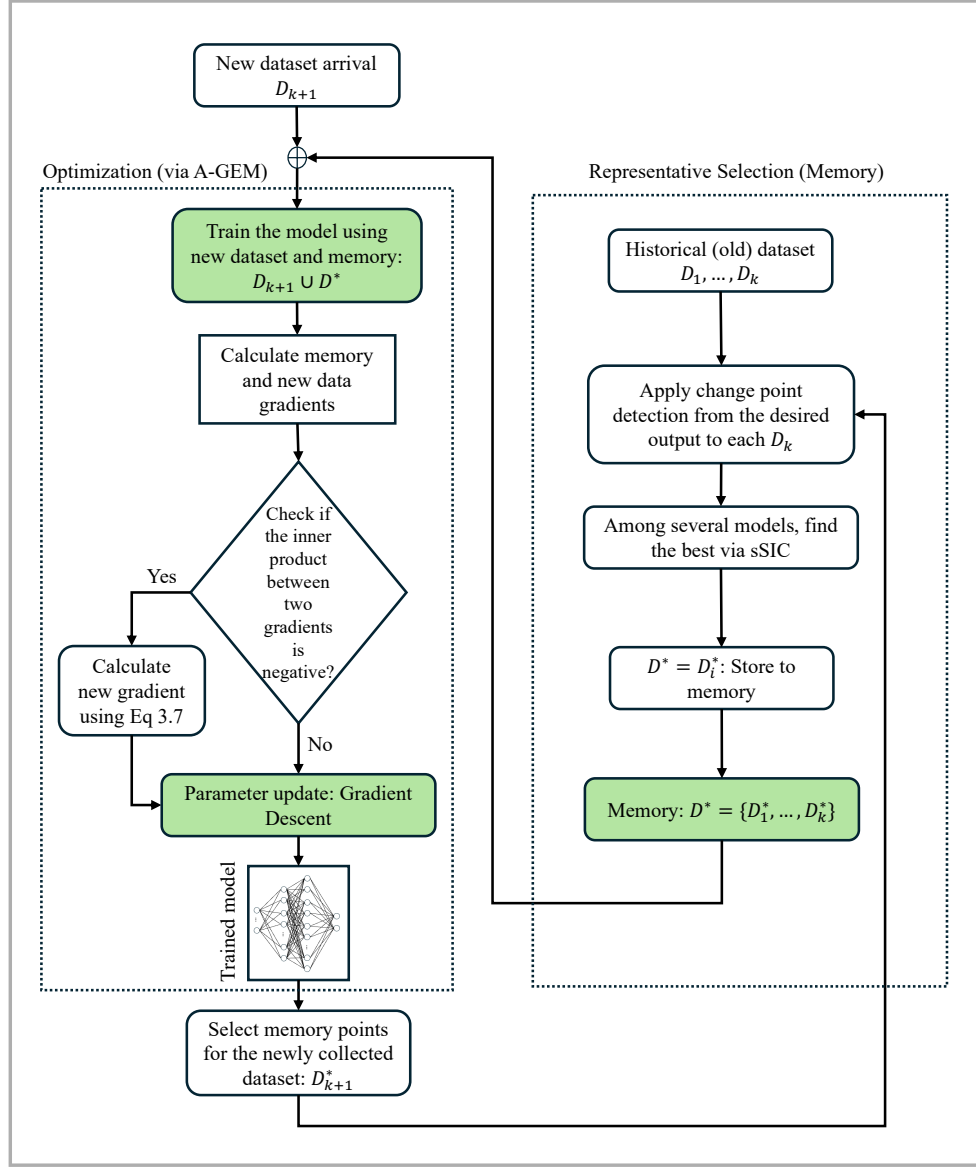


Figure 3.5: Step-by-step procedure of the proposed method.

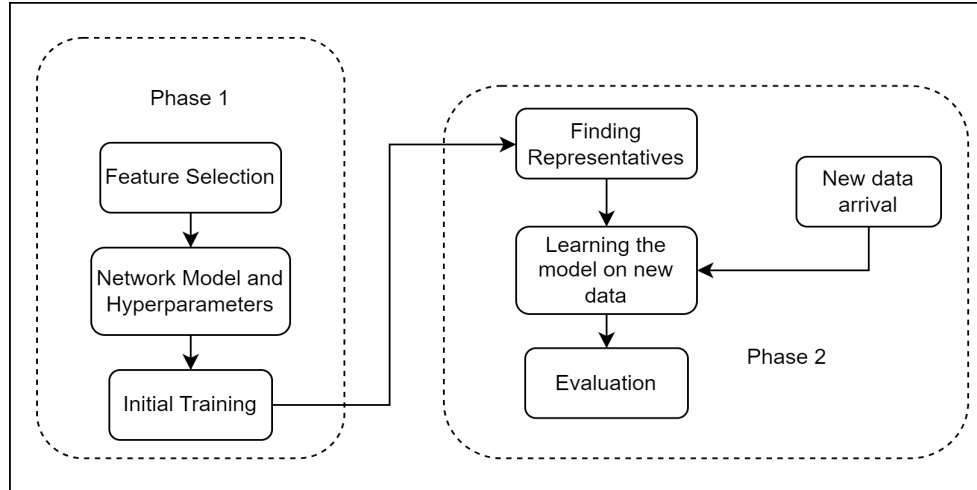


Figure 3.6: Step-by-step procedure of the evaluation.



Figure 3.7: Test vehicle used for experimental analysis.

lateral velocities, one additional maneuver from a new domain (D_2) is sequentially added to the model. Subsequently, the validation was performed after the addition of the new maneuver. Note that as previously specified, p is the number of input dimension of $\mathbf{x}_i$ in Eq. 3.1, which consists of different signal measurements used for this study. The specific number for p and the number of features used as input signals are discussed in the following subsection. Additionally, y_i is the specific output to be estimated (V_x or V_y) at each time step i . D_1 is the historical or old dataset, where the initial model will be trained on.

Table 3.1: Description of the dataset for estimation of longitudinal and lateral velocities.

Estimation	Data type	# of maneuvers	# of data
V_x	Training: D_1	22	52822
	Training: new domain D_2	1	2400
	Testing D_1	2	13800
V_y	Training: D_1	12	28812
	Training: new domain D_2	1	3000
	Testing D_1	2	4800

In subsections 3.5.2 and 3.5.3, the model’s performance in estimating the longitudinal velocity is evaluated, and in subsections 3.5.4 and 3.5.5, its performance in estimating the lateral velocity is assessed. Note that all experiments were conducted on a single system equipped with 11th Gen Intel(R) Core(TM) i7-11700, 16 GB of RAM (all experiments were tested on a single CPU with the same condition). Two evaluation metrics are used to assess the performance of the model on both the training and testing datasets. The first metric is the Mean Absolute Error (MAE), which quantifies the average absolute differences between the true (measured) values and the predicted values. It is computed as:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (3.13)$$

where y_i denotes the true (measured) value, $\hat{y}_i$ is the predicted value, and n is the total number of observations in a specific experiment. The second metric is the training time, which measures the duration required to train the neural network model at each learning stage, including both the initial and continual training phases for each method. This is a particularly valuable metric, as it highlights the benefits of continual learning in terms of reduced training time and computational cost.

3.5.2 Phase 1: Initial Training (Longitudinal Velocity)

Feature Selection

For estimation of longitudinal velocity, in addition to the last steps history of the predicted longitudinal velocity (the number of steps depends on the window size in the LSTM model, discussed in the next subsection), other relevant sensor measurements should also be considered in the input set using dynamic relations. To estimate longitudinal velocity, wheel speeds are often considered as the most relevant parameters, suggesting to inclusion of additional state measurement. Based on [70], wheel speeds, yaw rate, longitudinal and lateral accelerations, and steering angle are rational choices for the input set.

Network Model and Hyperparameters

As mentioned earlier, the approach to tackle the time series prediction problem involves using an LSTM model. A simple multilayer perceptron with fully connected layers is unable to fully capture nonlinear relationships between inputs and output states in a time-dependent scenario. To address this, a recurrent neural network (RNN) with feedback connections is developed. Long Short-Term Memory (LSTM) networks, a special class of RNNs, are specifically designed for time series and data stream prediction tasks. Figure 3.8 shows a detailed representation of an LSTM unit, which comprises three gates: the input gate, the forget gate, and the output gate. The input gate controls how much of the new input should be stored in the cell state by considering both the new input and the previous hidden state. The forget gate determines how much of the previous memory should be discarded. The output gate regulates how much of the cell state is exposed as the hidden state. In each LSTM cell, the following operations are performed:

$$i_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \quad (3.14)$$

$$f_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \quad (3.15)$$

$$o_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \quad (3.16)$$

$$g_t = \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \quad (3.17)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (3.18)$$

$$h_t = o_t \odot \tanh(c_t) \quad (3.19)$$

where $\odot$ denotes element-wise multiplication, $\sigma(\cdot)$ is the sigmoid activation function, and $\tanh(\cdot)$ is the hyperbolic tangent function. The vectors i_t , f_t , o_t , and g_t represent the

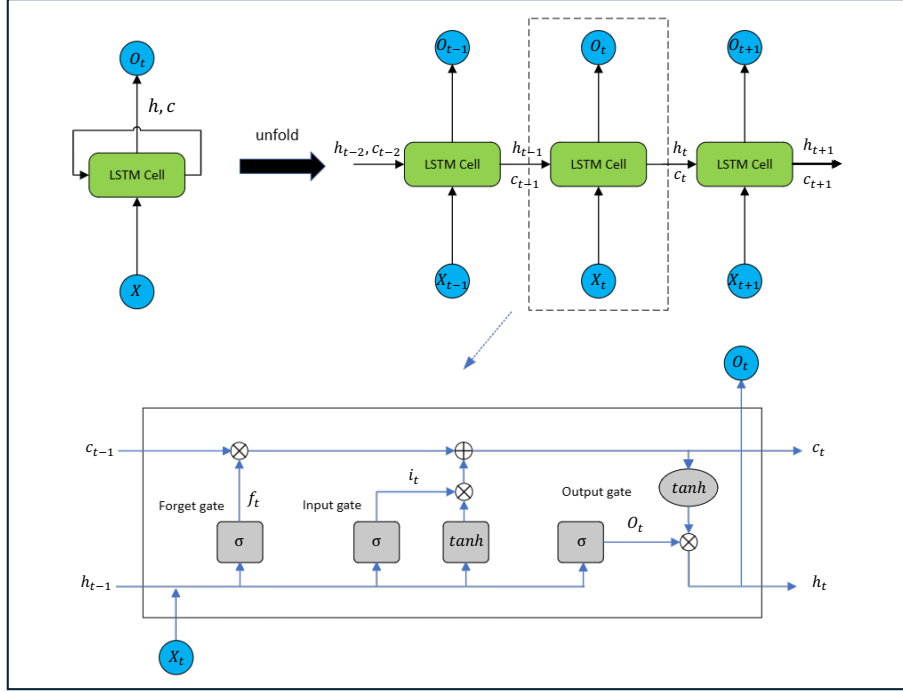


Figure 3.8: Structure of an LSTM cell with input, forget, and output gates.

input gate, forget gate, output gate, and candidate cell state, respectively. The matrices $\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo}$, $\mathbf{W}_{xc}$, $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho}$, and $\mathbf{W}_{hc}$ are learnable weight parameters, and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, $\mathbf{b}_c$ are their corresponding biases.

The hyperparameters for this model are detailed in Table 3.2. The input layer of the neural network consists of 8 previously chosen features. The initial node in each layer functions as the bias node. The output layer features a linear activation function, aggregating values from the preceding layers. The number of units in the hidden layers (h) is determined through a combination of dataset size considerations and experimentation. A learning rate of 0.01 is applied for the initial batch, which contains a large dataset, while a learning rate of 0.001 is used for the sequential batches. This choice of a lower learning rate for sequential batches is made to mitigate the impact of large gradients that often arise during the initial stages of training in continual learning scenarios. Also, the biased first and second moment of the gradients in the Adam optimizer [71] (m_i and v_i) are set to 0.9 and 0.95, respectively. The loss function used to optimize the neural network is a mean squared error (MSE), commonly used for regression tasks.

The behavior of each input feature can vary significantly across its minimum and max-

Table 3.2: The specifications of hyperparameters of the neural network for estimation of V_x .

Hyperparameter	Description	Value
l	Number of hidden layers	1
h	Number of hidden units	128
w	Window size	20
m_i	Biased first moment (Adam optimizer)	0.9
v_i	Biased second moment (Adam optimizer)	0.95
α	Learning rate for initial training	0.01
α'	Learning rate for continual learning	0.001

imum values. To ensure that these features are on a consistent scale, data normalization is applied before incorporating them into the neural network. However, data normalization presents challenges in the context of continual learning methods, as future data is unknown. One approach is to normalize the data based on the initial dataset. In this study, the initial dataset is extensive and includes a wide range of vehicle data, making it a valuable reference for data normalization. There are several data normalization techniques that can be used for neural networks. One common approach is *min-max normalization*, in which each input feature is scaled to a value within the range of 0 and 1. The normalized value of an input feature can be calculated using the following formula:

$$x_{\text{norm}}^{(i)} = \frac{x^{(i)} - x_{\min}^{(i)}}{x_{\max}^{(i)} - x_{\min}^{(i)}} \quad (3.20)$$

where $x^{(i)}$, $x_{\min}^{(i)}$, $x_{\max}^{(i)}$, and x_{norm} are the original input, the minimum and maximum values of the i th input feature, and the normalized input, respectively. The minimum and maximum values observed in the initial dataset are used as the reference for normalizing future data.

Training the model on the initial dataset

The first 22 maneuvers listed in Table 3.1 are used as the initial dataset to estimate V_x . It is called D_1 . Note that all training maneuvers in D_1 derived from experiments in the dry road. The training process on the initial dataset is terminated when the MSE of the predictions falls below a predefined threshold value (e.g., 0.05). The results of the training, including the Mean Absolute Error (MAE), along with the training time in seconds, are presented in

Table 3.3: Estimation results of V_x on training dataset after initial training.

	Total Size	epochs	MAE (km/h)	Training Time (s)
Initial Dataset	52822	365	0.1888	382.2

Table 3.4: Estimation results of V_x on test dataset after initial training (D_1).

Maneuver	Description	Maneuver time (s)	MAE
Test 1	Normal driving on the road	45	0.1570
Test 2	Step Steer	12	0.2893

Table 3.3. Two maneuvers are reserved for testing purposes which were previously derived from experiments tested on dry roads (the same condition as the training data). Table 3.4 and Figure 3.9 presents the estimation results of V_x obtained from the designed model on the two test datasets.

3.5.3 Phase 2: Continual Learning (Longitudinal Velocity)

Continual Learning Scenarios

An additional maneuver that is a distinct experiment from those used in the pre-trained model (introducing a new data domain) is introduced, and is called as D_2 . This maneuver was performed on a wet road and initially resulted in poor estimations by the neural network before being incorporated into the model. Figure 3.10 shows the estimation result of the longitudinal velocity on the new maneuver using the previously learned model. Maneuvers executed on wet surfaces for the latest data batches have been chosen. Consequently, less accurate estimations is anticipated, given that the model was initially trained on dry road conditions, and it must now adapt to novel environments. As depicted in the figure, poor estimation is observed in certain locations, indicating the need to train the existing model on the new task to improve its estimation accuracy.

CL Strategies

The proposed continual learning method is compared with four other strategies:

Batch: Retraining all neural network parameters, with all previous and newly collected data (baseline or upper bound). it is called as Batch or Joint training mode as suggested in [72].

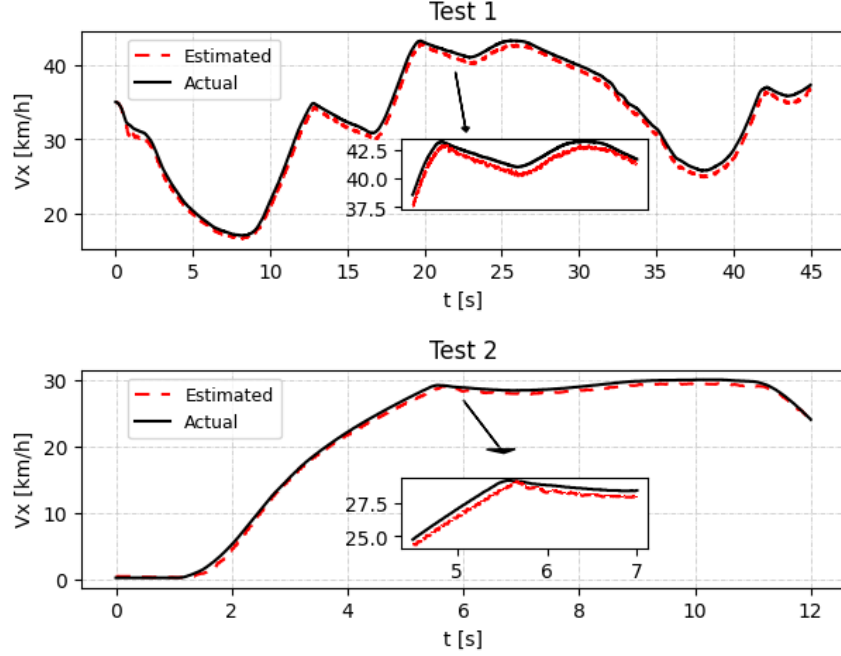


Figure 3.9: Estimation of V_x on the two test maneuvers after the initial training (D_1).

None: Only train the model on the new maneuver while ignoring the previous dataset (lower bound).

A-GEM [7]: The original A-GEM algorithm, where the memory points are selected based on the random sampling of the previous task.

SI [29]: This algorithm regularizes the loss function to avoid catastrophic forgetting by freezing important weights of the neural network, by measuring the sensitivity of each parameter.

Figure 3.11 shows the estimation of V_x on the new maneuver after introducing it to the model using ReSeleCT. The performance of the neural network in estimating the new, initial, and test maneuvers using all five strategies are presented in Table 3.5 and Table 3.6. To have a fair comparison, for both the proposed method and the A-GEM the same number of memory points were considered (total of 326 memory points for initial dataset D_1). As shown in Table 3.5, ReSeleCT offers a significant reduction in computation time while maintaining low prediction error in estimating the newly added data to the model. Training the model on the new maneuver using the Batch mode, which involves retraining the model with old and new datasets, requires 78.21 seconds. In contrast,

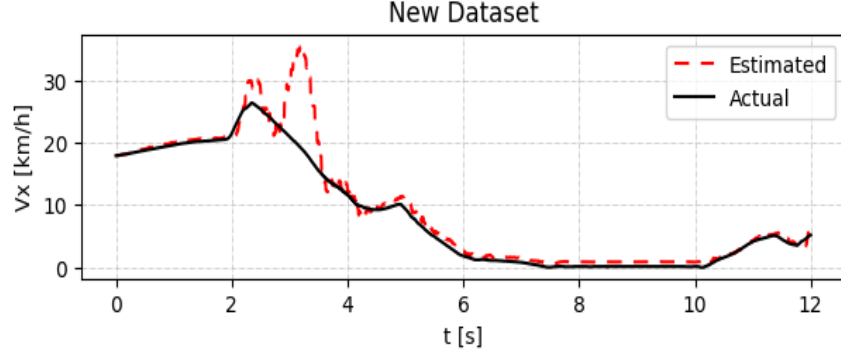


Figure 3.10: Estimation of V_x on the new data, prior to training the model on.

the proposed method only requires 6.47 seconds for training. The table reveals that SI exhibits the shortest computation time (after the None mode), leveraging its memory-less model that only requires parameter importance calculation before training. Additionally, ReSeleCT demonstrates a training time comparable to the other memory-based training methods like A-GEM, with a slightly longer duration attributed to representative selection computations, although it is negligible.

Table 3.5: Estimation results of V_x on the new dataset (D_2) after training the model on.

Maneuver	Method	Training Time (s)	MAE (km/h)
D_2	Batch	78.21	0.57
	None	2.87	0.49
	SI	3.5	0.53
	A-GEM	5.32	0.55
	ReSeleCT	6.47	0.55

The power of Continual Learning (CL) strategies lies in their ability to retain previously acquired information. As depicted in Table 3.6, the estimation mean absolute error values on training and testing datasets are comparable with Batch mode. However, in None mode, where only the new data is used to train the model, significant forgetting occurs, leading to increased prediction error values compared to the other two methods. Analyzing the table, the MAE for old data highlights the superior performance of the proposed method in preserving historical information compared to the other popular CL methods (SI and A-GEM). In most cases, SI exhibits poor estimation, indicating significant forgetting. Furthermore, a comparison between the two memory-based approaches (ReSeleCT and A-

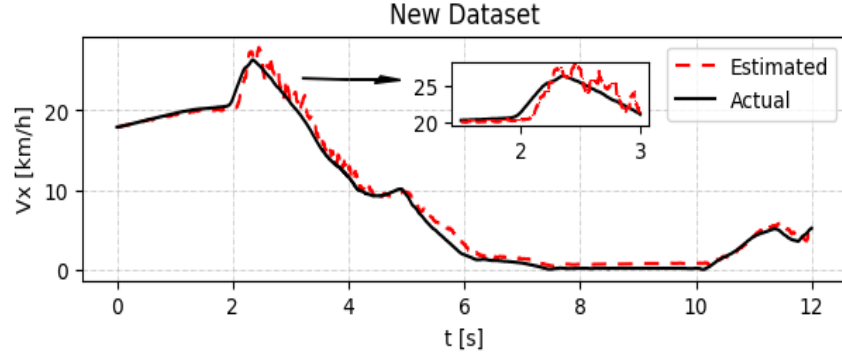


Figure 3.11: Estimation of V_x on the new dataset (D_2), after training the model on using ReSeleCT.

Table 3.6: Estimation Results (MAE) of V_x on training and testing maneuvers (D_1) after introducing the new dataset (D_2) to the model.

Method	Initial Dataset (D1)	Test 1	Test 2
Batch	0.19	0.69	0.58
None	1.23	8.81	3.82
SI	0.62	2.10	0.79
A-GEM	0.36	0.96	0.69
ReSeleCT	0.28	0.77	0.63

GEM) reveals that the model trained with A-GEM tends to forget information, particularly in the initial dataset (first column in the table 3.6) and test 1.

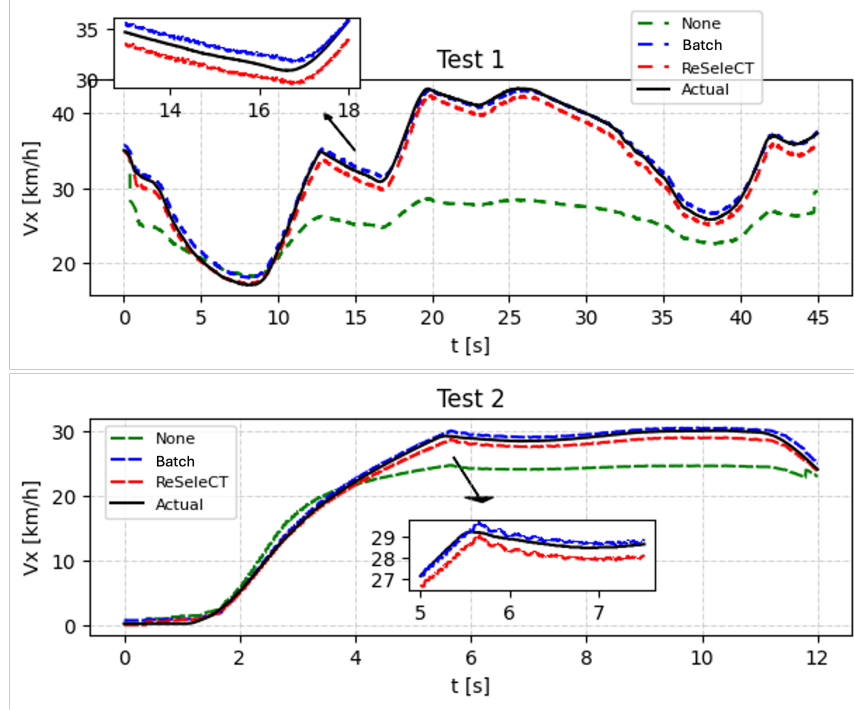


Figure 3.12: Estimation of V_x on the two test maneuvers after introducing the new dataset (D_2).

Figure 3.12 shows the longitudinal velocity estimation on the two test datasets, using Batch, None, and ReSeleCT. Focusing on areas with higher instances of forgetting in the None mode, it is observed that ReSeleCT reliably preserves locations where the function's pattern abruptly changes. In Test 1, specifically between seconds 14 and 18, where the None mode fails to retain information, the proposed approach effectively recovers the acquired knowledge. Similar occurrences are noted in Test 2, with intervals lasting from 5 to 7 seconds. Overall, the proposed method demonstrates comparable prediction accuracy in incorporating new data domains and retaining previously learned information compared to the Batch mode. Additionally, it offers a significant reduction in training time, a crucial advantage in CL strategies.

Figure 3.13 shows the longitudinal velocity estimation on the two test datasets, using SI, A-GEM, and ReSeleCT. As depicted in the figure, in Test 1 significant forgetting

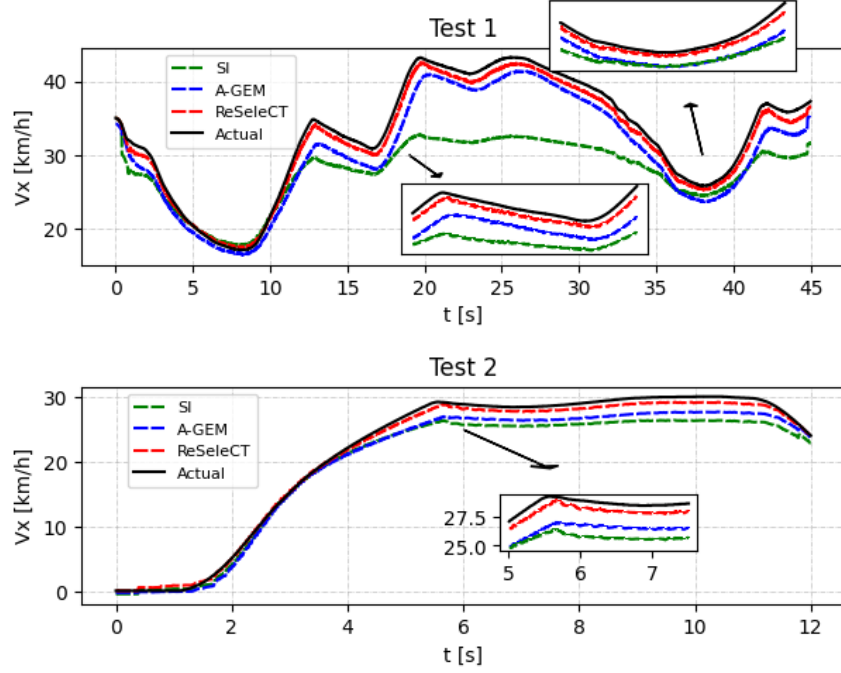


Figure 3.13: Comparison of the proposed method with other CL strategies in estimating V_x on testing data after introducing the new dataset (D_2) to the model.

occurs during time intervals 12-18, and 36-41 when using SI. Notably, in these instances, ReSeleCT’s ability to capture and preserve specific time sequences surpasses that of A-GEM, which selects memory points randomly. It is worth noting that forgetting in certain locations often cascades throughout the maneuver, given that LSTM relies on previously estimated velocities as inputs. Moreover, in Test 2, within the time window of 5 to 7, ReSeleCT demonstrates superior preservation of information, closely resembling the ground truth signal, while A-GEM struggles to retain knowledge from this timeframe.

3.5.4 Phase 1: Initial Learning (Lateral Velocity)

In this and the following sections, the effectiveness of the proposed method in continual learning of a vehicle’s lateral velocity is assessed. Given the similarities in the processes of estimating longitudinal and lateral velocities, to simplify the discussion and avoid repetition, additional explanations are skipped and the focus is solely on the results. Table 3.7 provides details on the hyperparameters used for lateral velocity estimation for the

initial model training. It is worth noting that for estimation of lateral velocity, based on [70], two-wheel speeds, yaw rate, lateral and longitudinal acceleration, and steering angle as the inputs, in addition to the last steps of the predicted lateral velocity were selected. Tables 3.8 and 3.9, and Figure 3.14 show the estimation results on the estimation of lateral velocity using the initial model.

Table 3.7: Hyperparameters of the neural network for estimating lateral velocity.

Hyperparameter	Description	Value
l	Number of hidden layers	1
h	Number of hidden units	128
d	Window size	25
m_i	Biased first moment (Adam optimizer)	0.95
v_i	Biased second moment (Adam optimizer)	0.99
α	Learning rate for initial training	0.005
α'	Learning rate for continual learning	0.001

Table 3.8: Estimation results of V_y on training dataset after initial training.

	Total Size	epochs	MAE (km/h)	Training Time (s)
Initial Dataset	28812	147	0.31	436

3.5.5 Phase 2: Continual Learning (Lateral Velocity)

The same scenarios for the continual learning phase are used. For the new dataset, a maneuver that was performed in a different condition than the initial dataset (D_1) was selected, representing a new data domain that needs to be adapted (D_2). Figure 3.15 shows the estimation result of the lateral velocity on the first new maneuver using the previously learned model. As depicted in the figure, poor estimation is observed in certain locations, indicating the need to train the existing model on the new task to improve its

Table 3.9: Estimation results of V_y on test datasets after initial training (D_1).

Maneuver	Description	Maneuver time (s)	MAE
Test 1	Double lane change	12	0.26
Test 2	Slalom	12	0.18

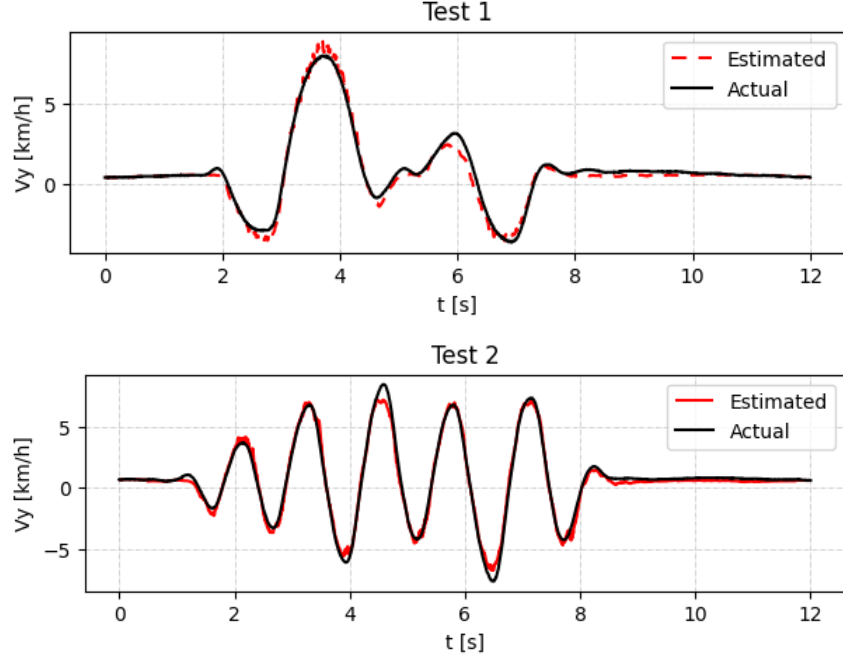


Figure 3.14: Lateral velocity estimation on two test maneuvers after the initial training (D_1).

estimation accuracy. Tables 3.10 and 3.11, and Figures 3.16 and 3.17 show the estimation results after adapting the model on the new dataset using different approaches (total of 275 memory points for initial dataset D_1 were considered for ReSeleCT and A-GEM). The findings demonstrate the efficacy of the proposed method in adapting to a new data domain while retaining previously acquired information, which is achieved through representative selection and the optimization technique used. Conversely, A-GEM exhibits significant catastrophic forgetting in the test data, indicating its inability to preserve prior knowledge.

3.5.6 On the sensitivity of the number of memory points in estimation error

To assess the robustness and efficiency of ReSeleCT, a sensitivity analysis is performed focused on the number of memory points, a critical parameter in ReSeleCT. As previously discussed, NOT method enables the automatic selection of the optimal number of memory points from historical data using the Schwarz Information Criterion (SIC) as the model

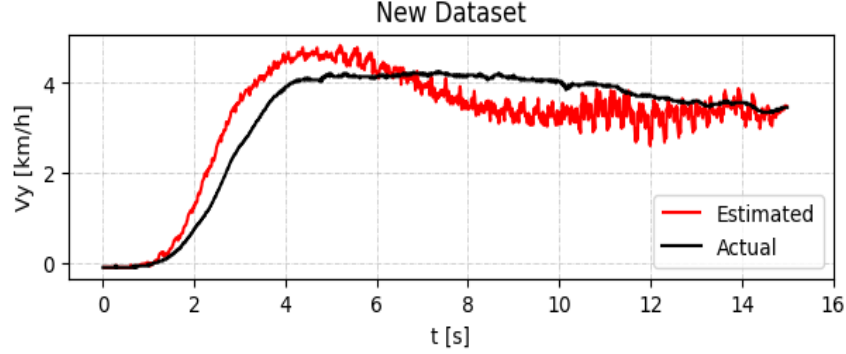


Figure 3.15: Estimation result on the new task, prior to training the model on.

Table 3.10: V_y estimation results on new maneuver (D_2) after training the model on.

Maneuver	Method	Training Time (s)	MAE (km/h)
D_2	Batch	81.1	0.22
	None	4.6	0.20
	SI	4.9	0.20
	A-GEM	7.1	0.21
	ReSeleCT	7.8	0.21

selection strategy. SIC uses a specialized cost function to identify the model with the least error while minimizing the number of selected points. This analysis is intended to provide insights into the impact of varying the number of memory points on the performance of ReSeleCT and how the most suitable number of memory points are automatically determined. It is important to note that the model selection process in ReSeleCT using the NOT approach is highly efficient, as demonstrated in Table 3.5, with only a negligible difference in computational time compared to A-GEM, where the selections are random and manual. Different experiments were performed by varying the number of memory points captured from the initial dataset D_1 and their impact on the model's performance were observed in terms of mean absolute error (MAE) of estimation of longitudinal velocity on training and testing data for the scenario that presented. The number of memory points was varied in increments. Figure 3.18 and Table 3.12 show the estimation error (MAE) of estimating V_x after introducing the new domain using ReSeleCT. The marked point in the figure is the automatically selected number of points using the proposed method, while the rest of the values are set manually as an independent experiment. As shown in the figure and the table, the automatically selected number of memory points (326) is the most rational choice,

Table 3.11: Estimation Results (MAE) of V_y on Training and testing maneuvers after introducing the new maneuver (D_2) to the model.

Method	Initial Dataset (D1)	Test 1	Test 2
Batch	0.21	0.27	0.18
None	2.42	1.2	0.95
SI	0.49	0.37	0.45
A-GEM	0.37	0.34	0.27
ReSeleCT	0.27	0.30	0.24

as the error reduction becomes marginal after this point for estimation of V_x for the initial dataset as well as the two test sets. This indicates that the selected number of memory points effectively balances performance improvement and memory storage efficiency.

Table 3.12: Memory Points vs. MAE of estimation of V_x for initial dataset and testing sets

Memory Points	MAE (Initial)	MAE (Test 1)	MAE (Test 2)
50	0.41	1.04	0.80
100	0.34	0.97	0.72
150	0.30	0.84	0.70
200	0.29	0.81	0.65
250	0.2870	0.78	0.64
300	0.2830	0.77	0.63
326	0.28	0.75	0.62
350	0.279	0.75	0.6180
400	0.278	0.7450	0.6150
450	0.277	0.7430	0.6140
500	0.276	0.7428	0.6138

3.5.7 Comparison of the performance after introducing multiple domains

So far, the performance of different methods has been evaluated under the addition of a single new maneuver, which introduces new information for estimating both longitudinal and lateral velocities. However, it is also important to assess whether the performance of various models—particularly the proposed one—remains consistent when more than

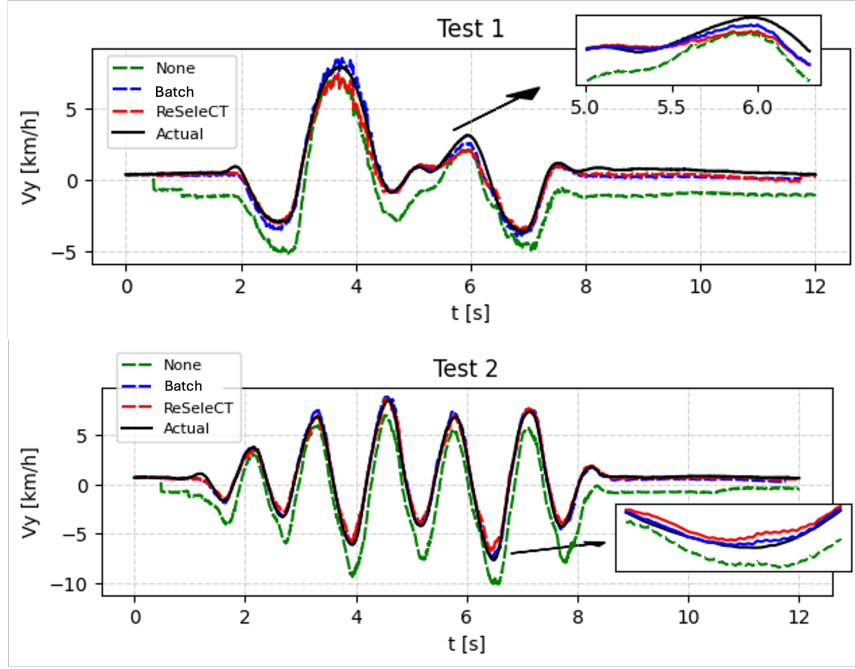


Figure 3.16: Longitudinal velocity estimation on two test maneuvers after the introducing the first new domain (D_2).

one new maneuver is introduced. To this end, in this subsection, the performance of different methods under the sequential addition of multiple maneuvers are evaluated, each providing novel information. Specifically, two key aspects were examined: (1) the extent to which the addition of new information leads to forgetting of previously learned knowledge from the initial dataset (D_1), and (2) the training time required at each stage. For this evaluation, in addition to the first and second domains discussed in the previous section, two additional maneuvers corresponding to domains D_3 and D_4 are introduced. Table 3.13 summarizes the specifications of each maneuver used in as an extension to Table 3.1. Note that, as previously described, the initial dataset in D_1 and the first new maneuver in D_2 are identical to those used in the prior experiments.

The first metric used to evaluate the performance of different models under the addition of multiple domains is the mean absolute error (MAE) of longitudinal and lateral velocity estimations on the initial dataset, measured after training the models on the first (D_2), second (D_3), and third (D_4) new maneuvers. An ideal model should exhibit zero forgetting, meaning the estimation error on the initial dataset remains unchanged after introducing each new maneuver. Figures 3.19 and 3.20 present the estimation errors for

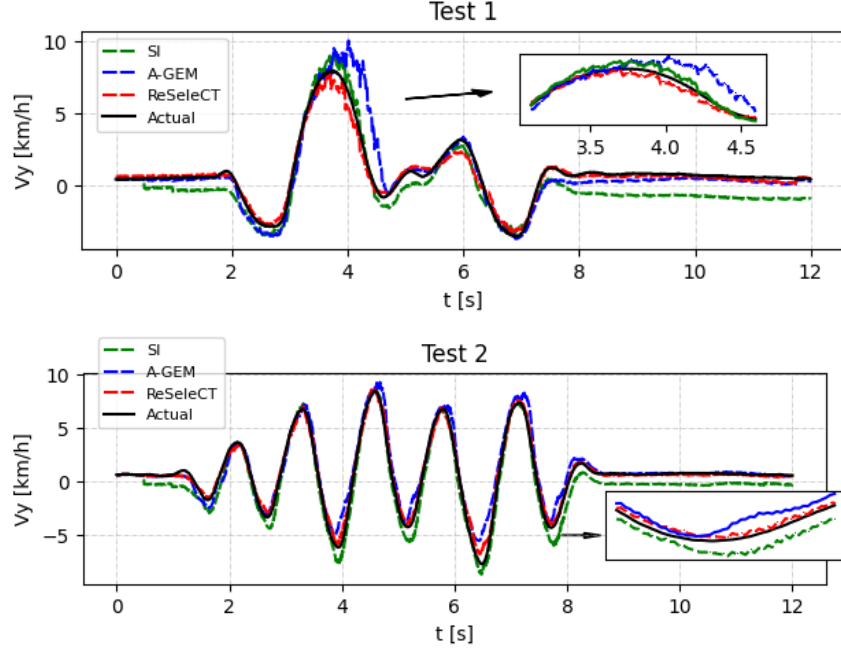


Figure 3.17: Comparison of the proposed method with other CL strategies in estimating V_y after introducing the new dataset (D_2) to the model.

longitudinal and lateral velocities, respectively, after each maneuver is added. The same model architectures as in the previous subsections are used for consistency. As shown in figures 3.19 and 3.20, for both longitudinal and lateral velocity estimation, the proposed method, ReSelecT, closely matches the performance of the Batch model (serving as the upper-bound baseline), even after the inclusion of the second and third maneuvers. In contrast, A-GEM fails to preserve the knowledge learned from the initial dataset, leading to increased estimation errors due to forgetting. This issue is even more pronounced with the SI method, where the degree of forgetting worsens as additional maneuvers are introduced. Finally, the None method, where the model is trained only on the new maneuver without any continual learning strategy, results in complete forgetting of previously learned information, leading to significantly higher estimation errors on the initial dataset.

Another important metric in evaluating the efficiency of continual learning scenarios, especially under the addition of multiple domains, is the training time required for each model to acquire new information. This becomes increasingly critical as the volume of training data grows with the introduction of new maneuvers, emphasizing the need for scalable continual learning approaches. Figures 3.21 and 3.22 illustrate the training time

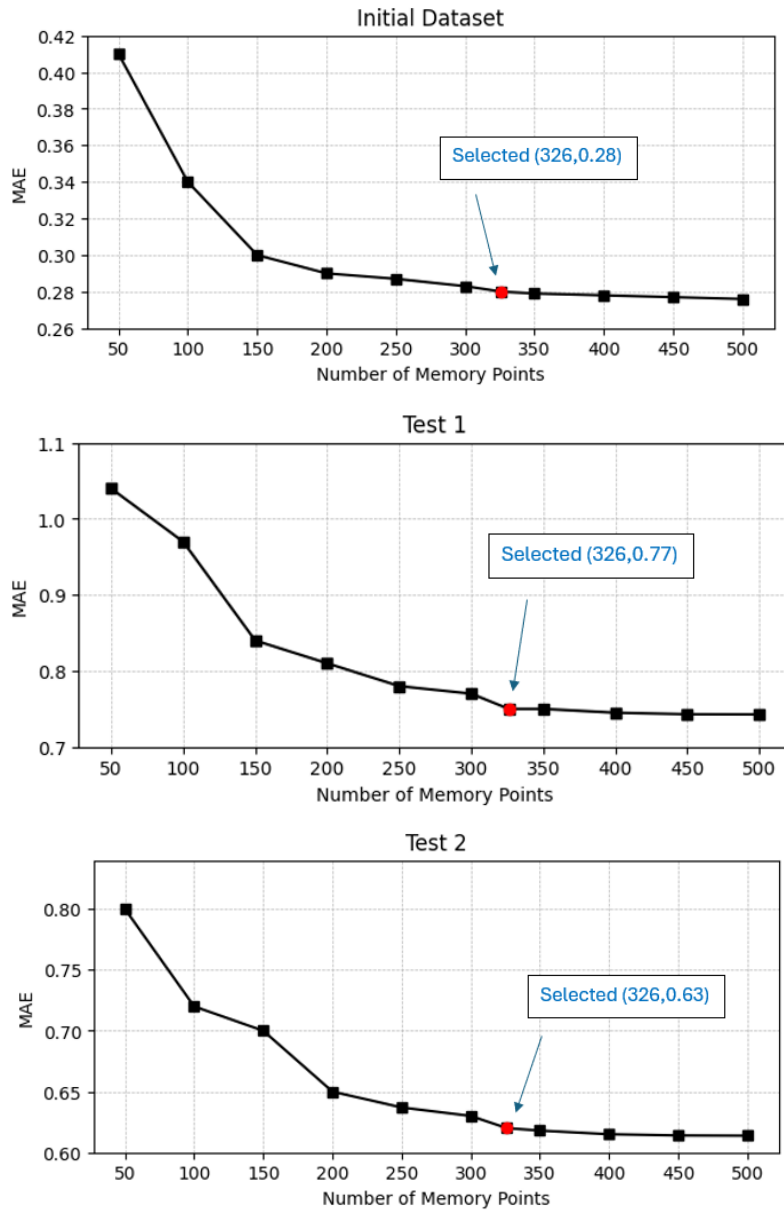


Figure 3.18: Sensitivity of the number of memory points on MAE for estimation of V_x after introducing the new domain using ReSelecT.

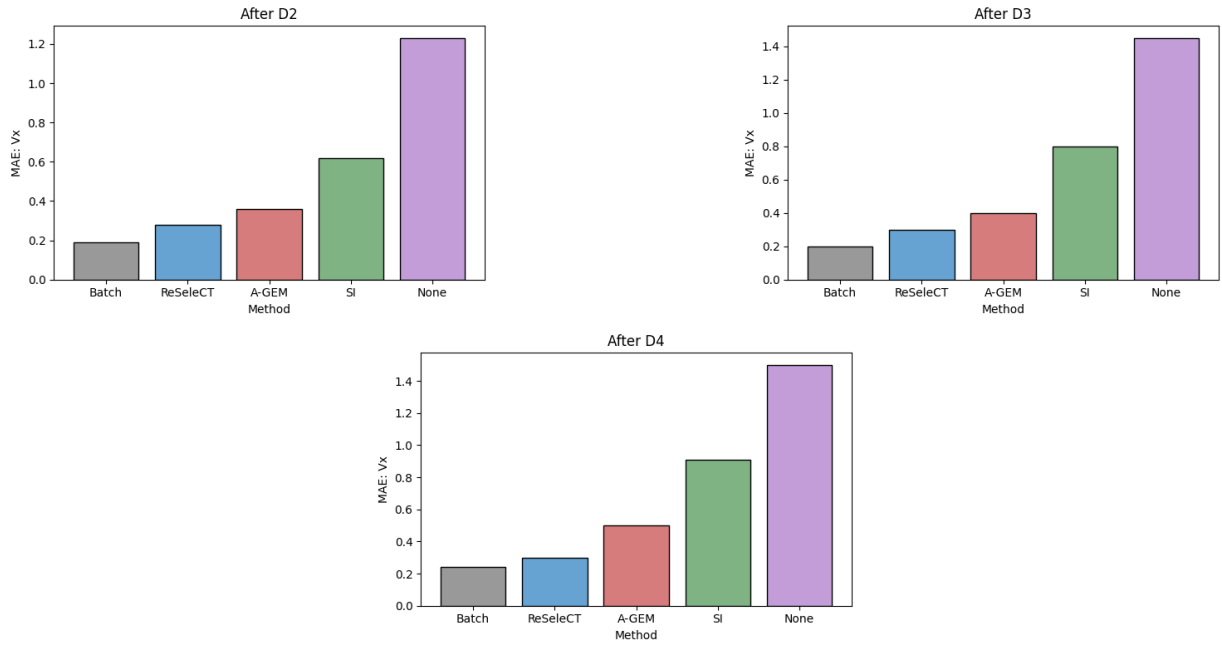


Figure 3.19: Estimation error of V_x for initial dataset after introducing each maneuver.

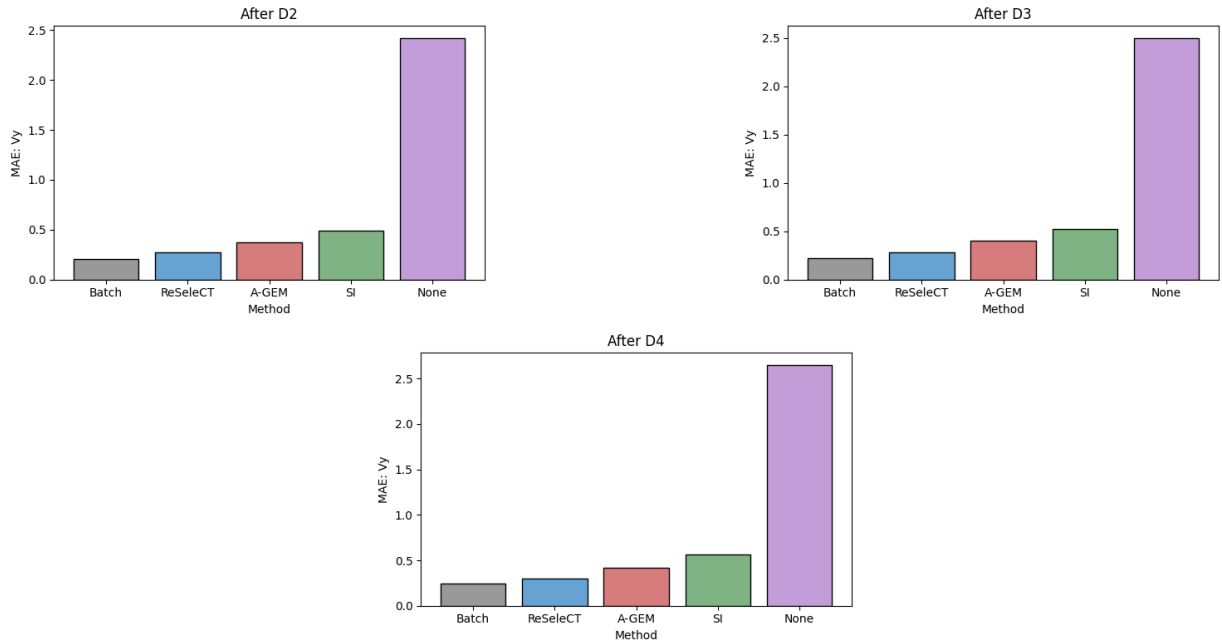


Figure 3.20: Estimation error of V_y for initial dataset after introducing each maneuver.

Table 3.13: Description of the extended dataset for estimation of longitudinal and lateral velocities under multiple addition of domains.

Estimation	Data type	# of maneuvers	# of data
V_x	Training: new domain $D3$	1	3200
	Training: new domain $D4$	1	2400
V_y	Training: new domain $D3$	1	2800
	Training: new domain $D4$	1	2400

of each method when incorporating new maneuvers. As shown, for both longitudinal and lateral velocity estimation, the Batch method requires significantly more training time, as it retrains the model using all historical data combined with the new data. This time demand increases substantially with each additional maneuver due to the cumulative growth of the dataset. In contrast, all other methods exhibit relatively consistent training times. Among them, the proposed ReSeleCT method and A-GEM incur slightly higher training times due to memory usage; however, this difference is negligible when compared to the substantial overhead of the Batch retraining strategy. Overall, the results demonstrate that the proposed method significantly reduces training time while maintaining comparable estimation accuracy to the baseline Batch method. This highlights the effectiveness of ReSeleCT in providing efficient and scalable continual learning without sacrificing performance. This balance between computational efficiency and predictive accuracy is crucial for real-time or resource-constrained applications. Therefore, ReSeleCT presents a practical and scalable solution for continual learning in sequential estimation tasks.

3.6 Summary

In this chapter a new approach for training a neural network model for regression problems in continual learning scenarios called ReSeleCT is introduced, focusing on applications in vehicle state estimation. The algorithm works based on addressing the two main challenges regarding continual learning. First, a strategy to effectively select the best data for memory data represent the historical data, and then an effective optimization strategy that integrates the memory points and the new coming data for new training. The findings of this study demonstrate that the proposed model not only effectively incorporates new data while maintaining previously acquired information, but also offers the advantage of low computational cost by removing a significant amount of historical data and only saving informative points for the new learning phase. The application in vehicle state estimation

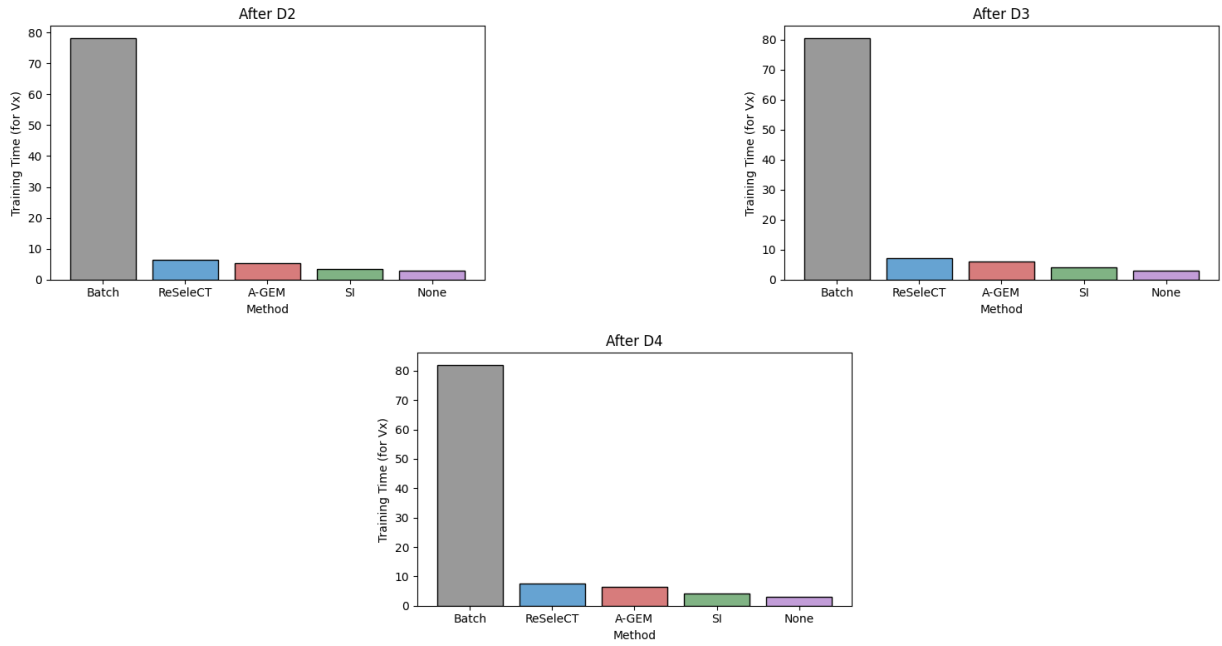


Figure 3.21: Training time for each model on Vx after introducing new maneuvers.

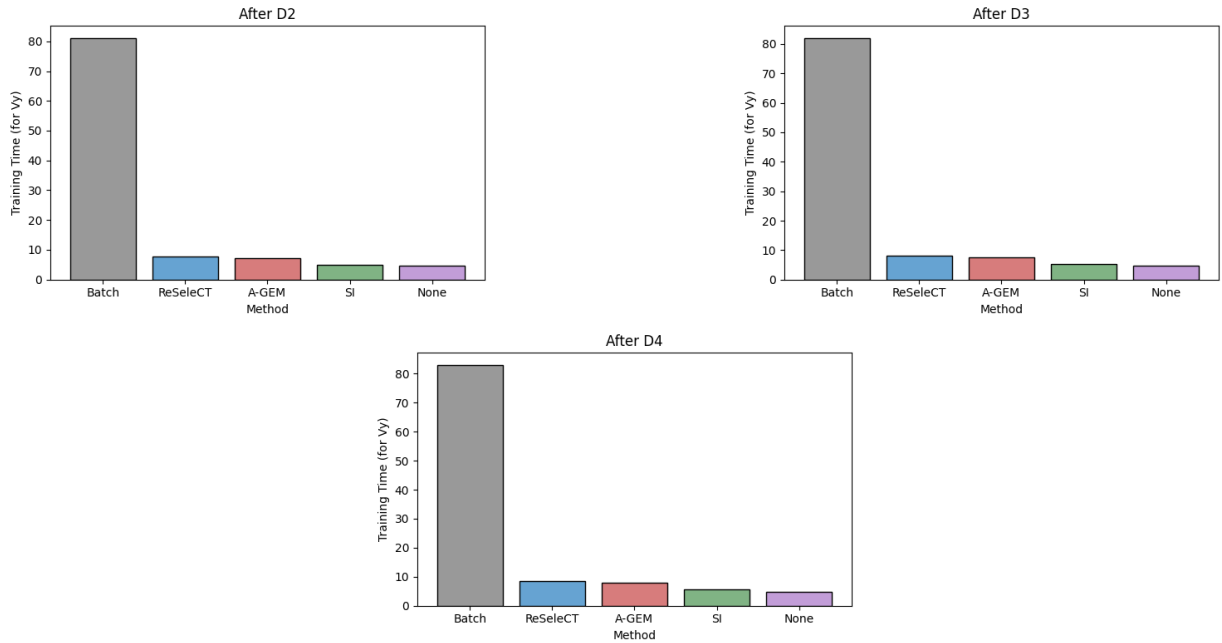


Figure 3.22: Training time for each model on Vy after introducing new maneuvers.

shows the effectiveness of the proposed approach in large datasets, particularly in vehicle datasets.

Chapter 4

Developing the Continual Learning Framework in Multi-output Time Series Regression Tasks

4.1 Introduction

In this chapter, continual learning in time series scenarios is further investigated and is extended to multivariate output variables scenarios. Applying the representative selection method introduced in Chapter 3 independently to each output often leads to redundancy; here, this limitation is addressed. Also a refined version of the A-GEM optimization algorithm is presented to better mitigate catastrophic forgetting in the target application. A comprehensive experimental analysis is conducted to evaluate the performance of the proposed method, comparing it against other state-of-the-art continual learning approaches. For the neural network architecture, an encoder-decoder Transformer model well-suited for the state estimation task is used. Unlike in the previous chapter, where input selection was based on dynamic relationships, a thorough feature selection using sensitivity analysis of input-output dependencies is performed. Finally, a detailed uncertainty quantification is included to assess the reliability of the neural network’s predictions. Note that throughout this chapter, the proposed approach is referred to as EM-ReSeleCT (an Efficient Multivariate Representative Selection for Continual learning in Time-series scenarios) [73].

4.2 Formulating the Problem in Multi-output Regression

Consider a stream of observations denoted as $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1, 2, \dots$, where $\mathbf{y}_i \in \mathbb{R}^q$ represents the desired output or response vector, and $\mathbf{x}_i \in \mathbb{R}^p$ is the input feature vector. The objective is to approximate an unknown $\mathbb{R}^q$ valued function f based on a dataset D of observations $(\mathbf{x}_i, \mathbf{y}_i)$. Assume that for a given integer d , which indicates the order of temporal dependency between $(\mathbf{x}_i, \mathbf{y}_i)$ s and $i > d$,

$$\mathbf{y}_i = f(\mathbf{x}_i, H_i(D)) + \epsilon_i, \quad (4.1)$$

where $H_i(D) = \{(\mathbf{x}_j, \mathbf{y}_j) \in D; i - d \leq j < i\}$, and $\epsilon_i \in \mathbb{R}^q$ is the additive noise. As previously mentioned, neural networks have proven to be highly effective in estimating the function f for time series problems. By adopting the neural network approach, we inherently assume that f in (4.1) can be adequately estimated by a neural network parameterized by the vector $\boldsymbol{\theta}$. Therefore, use $f(\cdot; \boldsymbol{\theta})$ is used in place of f , effectively reducing the estimation of f to estimating the parameter vector $\boldsymbol{\theta}$. Let $\ell_i(\boldsymbol{\theta}) = \|\mathbf{y}_i - f(\mathbf{x}_i; \boldsymbol{\theta})\|^2$ represent the squared error loss for the i -th observation, where $\|\cdot\|$ is the l_2 norm of the resulting vector. To estimate $\boldsymbol{\theta}$, it is common to minimize the empirical risk:

$$R(\boldsymbol{\theta}, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \ell_i(\boldsymbol{\theta}). \quad (4.2)$$

where $|D|$ represents the total number of data points in dataset D . Suppose there are k datasets for a given experiment, each comprising time series data structured as in (4.1):

$$\begin{cases} D_1 = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_{n_1}, \mathbf{y}_{n_1})\} \\ D_2 = \{(\mathbf{x}_{n_1+1}, \mathbf{y}_{n_1+1}), \dots, (\mathbf{x}_{n_2}, \mathbf{y}_{n_2})\} \\ \vdots \\ D_k = \{(\mathbf{x}_{n_{k-1}+1}, \mathbf{y}_{n_{k-1}+1}), \dots, (\mathbf{x}_{n_k}, \mathbf{y}_{n_k})\} \end{cases} \quad (4.3)$$

Here, $n = \sum_{i=1}^k n_i$ represents the total number of data points in the combined dataset. In a continual learning setting, datasets from different domains denoted as $D = D_1 \cup D_2 \cup \dots \cup D_k$, are not available all at once. Instead, new datasets $(D_2, \dots, D_k)$ arrive sequentially, requiring the model to update with new data while retaining previously learned information. In this regard, after training on an initial dataset D_1 , the idea is to select a representative

subset of m_1 landmarks, $m_1 \ll n_1$, denoted as D_1^* , to retain in memory. Let us consider the selected landmarks from dataset D_1 as:

$$D_1^* = \{(\mathbf{x}_1^*, \mathbf{y}_1^*), \dots, (\mathbf{x}_{m_1}^*, \mathbf{y}_{m_1}^*)\}. \quad (4.4)$$

This subset is used in conjunction with newly received data D_2 to update the model parameters $\boldsymbol{\theta}$ by minimizing the empirical risk while ensuring that the risk on the previously selected representatives does not exceed the previous risk:

$$\min_{\boldsymbol{\theta}} R(\boldsymbol{\theta}, D_2) \quad \text{s.t.} \quad R(\boldsymbol{\theta}, D_1^*) \leq R(\hat{\boldsymbol{\theta}}_1, D_1^*). \quad (4.5)$$

4.3 Refined A-GEM Optimization algorithm

As outlined in Chapter 3, an effective solution to the optimization problem in Eq. 4.5 is the average gradient of the episodic memory, which is detailed in Section 3.3. However, A-GEM places significant emphasis on the new gradient, which may result in issues when the new gradient is too small compared to the gradient of the memory in some specific training epochs [49]. To overcome this limitation, a simple modification of A-GEM is developed. This modified approach places emphasis on the gradient of the memory when the gradient of the new data is too small. To do so, when $\langle \tilde{g}, g_{k+1} \rangle \leq 0$, the new gradient projection is conditioned to the magnitude of the two gradients. If the magnitude of the new gradient is smaller than the magnitude of the memory gradient, instead of projecting the new gradient, the algorithm projects the gradient of memory to the nearest gradient that satisfies $\langle \tilde{g}, g_{k+1} \rangle \geq 0$:

$$\tilde{g} = \begin{cases} g_{k+1} - \frac{\langle g_{k+1}, g_k^* \rangle}{\|g_k^*\|_2^2} g_k^*, & \text{if } \|g_{k+1}\| \geq \|g_k^*\| \quad (\text{a}) \\ g_k^* - \frac{\langle g_{k+1}, g_k^* \rangle}{\|g_{k+1}\|_2^2} g_{k+1}, & \text{if } \|g_k^*\| > \|g_{k+1}\| \quad (\text{b}) \end{cases} \quad (4.6)$$

The second term in (4.6) ensures that, during certain training epochs where the new gradient is smaller than the memory gradient, greater emphasis is placed on the memory gradient. Figure 4.1 shows the directions of the new gradient and the gradient of the memory for the three scenarios specified. In both terms of Eq. (4.6), it can be shown that the inner product between the gradient of the new data and the gradient of the memory points is positive.

Proposition: After calculating $\tilde{g}$, the resulting gradient in the second term (b) of (4.6) and the memory gradient are non-negatively aligned:

$$\langle g_k^*, \tilde{g} \rangle \geq 0 \quad (4.7)$$

Proof:

$$\begin{aligned} \langle g_k^*, \tilde{g} \rangle &= \left\langle g_k^*, \left(g_k^* - \frac{\langle g_k^*, g_{k+1} \rangle}{\langle g_{k+1}, g_{k+1} \rangle} g_{k+1} \right) \right\rangle \\ &= \langle g_k^*, g_k^* \rangle - \frac{\langle g_k^*, g_{k+1} \rangle}{\langle g_{k+1}, g_{k+1} \rangle} \langle g_k^*, g_{k+1} \rangle \end{aligned} \quad (4.8)$$

$$= \|g_k^*\|^2 - \frac{\|g_k^*\| \|g_{k+1}\| \cos(\theta)}{\|g_{k+1}\|^2} \cdot \|g_k^*\| \|g_{k+1}\| \cos(\theta) \quad (4.9)$$

$$= \|g_k^*\|^2 (1 - \cos^2(\theta)) = \|g_k^*\|^2 \sin^2(\theta) \geq 0 \quad (4.10)$$

4.4 Multi-output Representative Selection in time series tasks

In the proposed approach in multi-output space, the focus is on identifying specific changes in the output function pattern for each dimension, as these locations are more informative for time series regression tasks and are particularly vulnerable to catastrophic forgetting. To do so, the proposed method (EM-ReSeleCT) applies change point detection methods to find specific landmarks based on each output variable and then applies an error-based filtering of proximal selected points in each output variable to find the best subset and avoid redundant selection. To begin, consider a multi-output setting of p input and q output variables, as in Eq. 4.1 where q represents the number of outputs to be estimated. Similar to Eq. 4.3, let D denote the dataset of n instances for the pair of input and output vectors, i.e.

$$D = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}, \quad (4.11)$$

where $\mathbf{x}_j = (x_{1j}, \dots, x_{pj})'$ and $\mathbf{y}_j = (y_{1j}, \dots, y_{qj})'$ are p - and q -dimensional vectors of inputs and outputs at time instance j , respectively. Additionally, for $i = 1, 2, \dots, q$, let $\mathcal{Y}_i = \{y_{i1}, y_{i2}, \dots, y_{in}\}$ represents the dataset corresponding to the i^{th} coordinate (variable) of the multi-output time series $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$. The objective is to identify representative

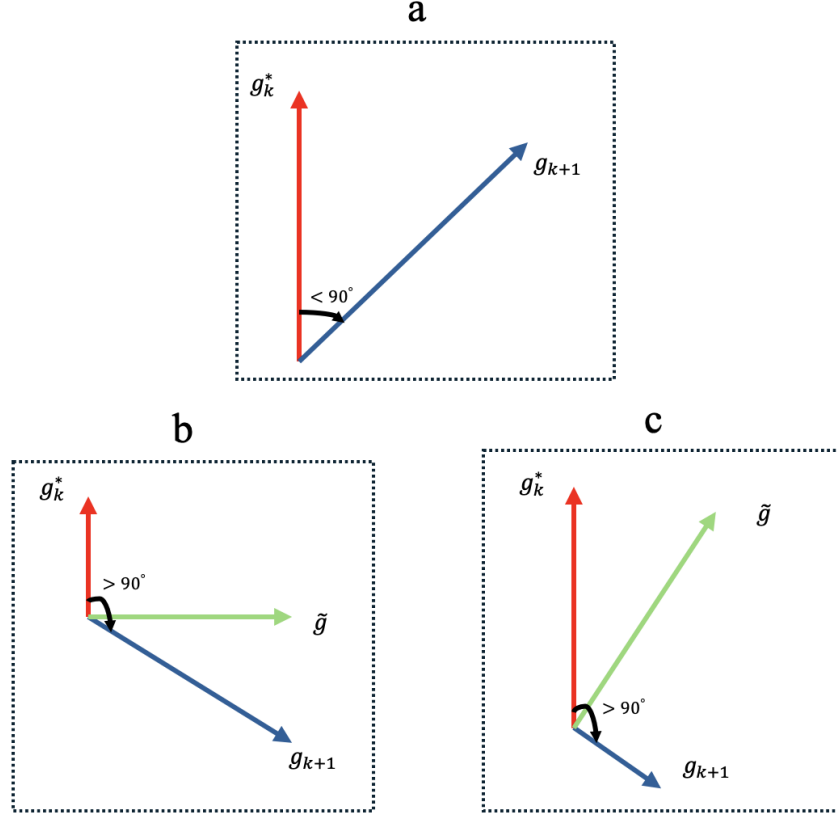


Figure 4.1: The position of the two gradients in different scenarios. (a): When the inner product is positive. (b): When the inner product is negative and the magnitude of the new gradient is larger. (c): When the inner product is negative, and the magnitude of the new gradient is smaller.

points from the dataset D that best capture the original dataset's structure. To achieve this, at the first stage, similar to the previous chapter, a change point detection method called the Narrowest-Over-Threshold (NOT) [69] is utilized to detect changes in the mean function μ_t of each output time series $\mathcal{Y}_i$. Let us denote the set of the change points by $\mathcal{Y}_i^*$ (performing over all dimensions of the output dataset in the historical dataset $\mathcal{Y}$). Therefore, the procedure begins by focusing on selecting a representative subset of the output data for each output variable individually. The NOT algorithm begins by partitioning the dataset $\mathcal{Y}_i$ into M_i randomly selected sub-samples. These sub-samples are determined based on observation indices, as depicted in Figure 4.2, where the interval $(s, e]$ consists of consecutive observations $y_{i(s+1)}, \dots, y_{ie}$. Assuming each sub-sample contains at

most one change point, the Gaussian Generalized Likelihood Ratio (GLR) test statistic is computed for potential change points within the interval $(s, e]$. If no change point is present, the likelihood function $L(y_{i(s+1)}, \dots, y_{ie}; \theta)$ is maximized to estimate the intercept and slope parameters $\eta = (b, m)$ of the linear mean function μ_t over the range $t \in (s, e]$. Conversely, if a change point occurs at $c \in (s, e]$, the mean function μ_t exhibits distinct intercepts and slopes in the sub-intervals $(s, c]$ and $(c, e]$, characterized by parameters η_1 and η_2 , respectively. The GLR test statistic, used to compare the likelihood of no change point versus the presence of a change point at c , is formulated as:

$$\mathfrak{R}_{(s,e]}^c(\mathcal{Y}_i) = 2 \log \left[\frac{\sup_{\eta_1, \eta_2} \{ L(y_{i(s+1)}, \dots, y_{ic}; \eta_1) L(y_{i(c+1)}, \dots, y_{ie}; \eta_2) \}}{\sup_{\eta} L(y_{i(s+1)}, \dots, y_{ie}; \eta)} \right] \quad (4.12)$$

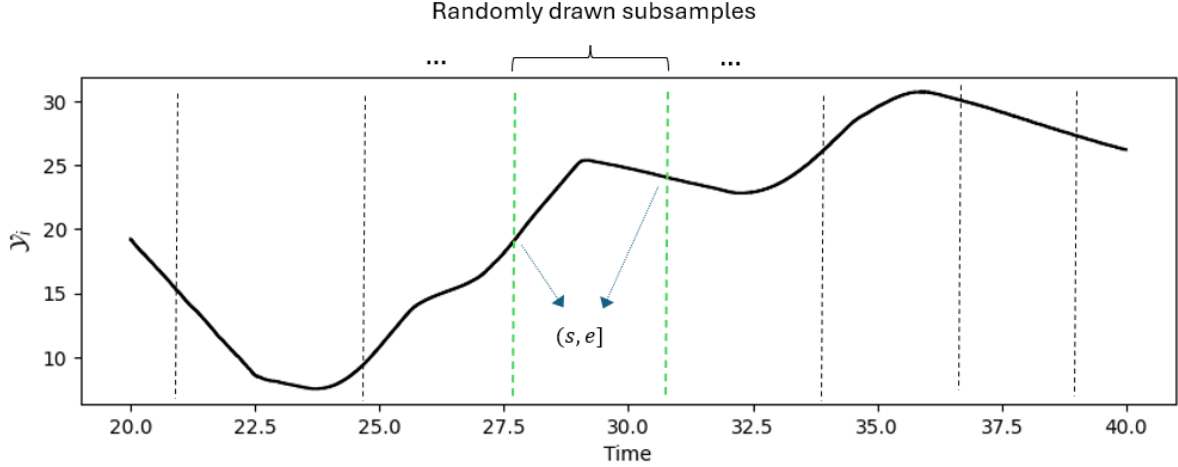


Figure 4.2: Representative selection on each dimension separately.

Applying the GLR test to each interval $(s, e]$, the NOT method estimates a change point by identifying the value of c that maximizes equation (4.12):

$$\mathfrak{R}_{(s,e]}(\mathcal{Y}_i) = \max_{c \in \{s+2, \dots, e-2\}} \mathfrak{R}_{(s,e]}^c(\mathcal{Y}_i) \quad (4.13)$$

After gathering all change points (landmarks) from each output dimension, i.e. $\mathcal{Y}_1^*, \mathcal{Y}_2^*, \dots, \mathcal{Y}_q^*$, the algorithm collects their corresponding value from the other output dimensions. This

results in q distinct sets, each corresponding to the selected change points based on one variable, along with the corresponding outputs from other dimensions:

$$\begin{cases} \mathcal{Y}_1^{**} = (\mathbf{y}_1^{(1)}, \dots, \mathbf{y}_{m_1}^{(1)}) \\ \mathcal{Y}_2^{**} = (\mathbf{y}_1^{(2)}, \dots, \mathbf{y}_{m_2}^{(2)}) \\ \vdots \\ \mathcal{Y}_q^{**} = (\mathbf{y}_1^{(q)}, \dots, \mathbf{y}_{m_q}^{(q)}) \end{cases} \quad (4.14)$$

where $\mathcal{Y}_i^{**}$ represents the set of representative points selected based on the i th output variable in the dataset and $\mathbf{y}_j^{(i)} = (y_{1j}^{(i)}, y_{2j}^{(i)}, \dots, y_{qj}^{(i)})$ is an observation vector in the dataset where the mean function of i th coordinate (or output variable) corresponding to j th change when only investigating the i th output variable. One potential solution is to identify change points independently for each output, followed by aggregating these points along with their respective inputs and outputs. However, this approach can result in inefficiencies due to the excessive selection of points. This inefficiency arises from the possibility of redundant point selection, as certain points may already have been chosen for a different output variable in a neighboring measurement. This problem becomes particularly pronounced in real-world state estimation applications, where the number of measurements is large, and there is minimal variation between consecutive points. Figure 4.3 illustrates this issue using the i th output time series dataset. In subplot (b), the blue point ($y_{i(k+1)}^{(i)}$) represents the change point detected based on the i th time series output variable, while the red point ($y_{i(g+1)}^{(j)}$) corresponds to a change point detected in another output variable. As shown in the figure, there is no significant difference between selecting either of these points, and selecting both would be redundant. However, this strategy does not apply to subplot (a), where selecting the red point ($y_{ig}^{(j)}$) and discarding the blue one ($y_{ik}^{(i)}$) would result in a significant error.

To address this, a proximal change point selection with minimum error-based filtering is proposed. The unique points from each representative set are concatenated. Thus, the concatenated representative set is given by:

$$\mathcal{Y}^{*c} = \{\mathcal{Y}_1^{**}, \mathcal{Y}_2^{**}, \dots, \mathcal{Y}_q^{**}\} \quad (4.15)$$

Let $\mathcal{Y}^{*c}$ represent the set of representative points obtained by concatenating all selected change points for the j th dataset. Now, consider the i th time series output variable in $\mathcal{Y}$

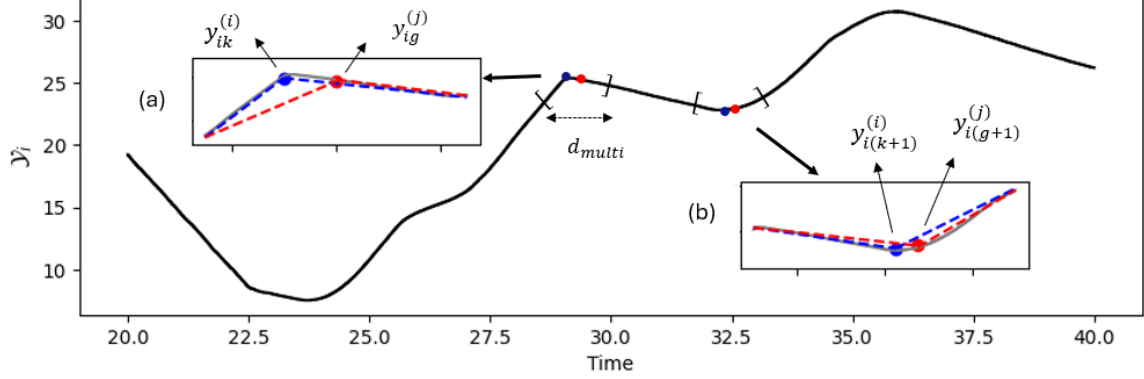


Figure 4.3: Multi-output representative selection by filtering based on error fit.

within dataset D . Based on eq. 4.15, the set $\mathcal{Y}_i^*$ is identified as the set of change points for this output variable. Suppose the j^{th} point in this set is denoted by y_{ij} .

It is assumed that within the defined distance d , only one change point is detected for the i^{th} output variable. This assumption holds for small distances. To evaluate the significance of this change point, the Generalized Likelihood Ratio (GLR) test is applied. Specifically, the GLR test is performed on the segment from $y_{i(j-\frac{d}{2})}$ (the start of the distance) to the selected point $y_{ij}^{(i)}$, and from this point to $y_{i(j+\frac{d}{2})}$ (the end of the distance):

$$\mathfrak{R}_{(s_1, e_1]}^{k, i}(\mathcal{Y}_i) = 2 \log \left[\frac{\sup_{\eta_1, \eta_2} \{L(y_{i(s_1+1)}, \dots, y_{ik}; \eta_1) L(y_{i(k+1)}, \dots, y_{ie_1}; \eta_2)\}}{\sup_{\eta} L(y_{i(s_1+1)}, \dots, y_{ie_1}; \eta)} \right] \quad (4.16)$$

where, $s_1 = j - \frac{d}{2}$ and $e_1 = j + \frac{d}{2}$, and $y_{ik}^{(i)}$ is the k^{th} change point detected based on the i^{th} output variable, and the notation k, i in eq. 4.16 represents the GLR test for k^{th} potential change point for output variable i . Now, for any change point detected in other output variables, if the corresponding value in the i^{th} variable falls within the corresponding interval, the GLR test is applied to the same interval as follows:

$$\mathfrak{R}_{(s_2, e_2]}^{g, j}(\mathcal{Y}_i) = 2 \log \left[\frac{\sup_{\eta_1, \eta_2} \{L(y_{i(s_2+1)}, \dots, y_{ig}; \eta_1) L(y_{i(g+1)}, \dots, y_{ie_2}; \eta_2)\}}{\sup_{\eta} L(y_{i(s_2+1)}, \dots, y_{ie_2}; \eta)} \right] \quad (4.17)$$

Note that the notations for Eqs. (4.16) and (4.17) matches the notations in Figure 4.3 in subplot (a). In other words, the GLR test is applied to the interval in subplot (a), one

for point y_{ig}^j and one for point y_{ik}^i . The error between the two GLR values is defined as:

$$E = \mathfrak{R}_{(e_1, s_1]}^{k,i}(\mathcal{Y}_i) - \mathfrak{R}_{(e_1, s_1]}^{g,j}(\mathcal{Y}_i) \quad (4.18)$$

The change point y_{ig}^j is removed from the representative set if the difference between the two GLR values in Eqs. (4.16) and (4.17) is below a predefined threshold. This is because its effect is negligible, and its neighboring point within the proximity can compensate for its absence. Conversely, if the GLR difference exceeds the threshold, removing y_{ig}^j would significantly impact the representation, so the point is retained in the set. After selecting the optimal representative set, it is stored in memory and used in conjunction with newly collected data during model training. This process efficiently mitigates the effects of catastrophic forgetting in continual learning models. Algorithm 2 shows the step-by-step procedure for selecting a representative set for memory-based continual learning in multi-output time series scenarios. After collecting the complete historical dataset from previous domains, a sequence-to-sequence Transformer model (with detailed discussion on its structure in the next section) is initially trained on the first dataset as a pre-trained model. For each historical dataset (D_j) in D , a multivariate change representative selection method is applied to identify the most representative subset of D_j , which consists of selected output points and their corresponding inputs. After gathering all these landmarks into D^* and receiving the new dataset D_{k+1} from the latest domain, the model is retrained using the combined dataset ($D^* \cup D_{k+1}$) with the modified A-GEM optimization algorithm to ensure least forgetting of the previously trained model. Following this continual training on the new dataset, the same representative selection process is applied to the new dataset, with the selected representatives stored for future learning tasks.

4.5 Evaluation and Testing

In this section, a comprehensive evaluation of the proposed method, compared to other continual learning approaches, is presented. The objective is to estimate both the longitudinal and lateral velocities of a vehicle using datasets collected through experiments conducted with an electric Equinox vehicle. Table 4.1 outlines the specifications of the dataset used in this study. Two sets of training data were used: an initial, relatively large dataset for training in the initial domain (D_1) and a new dataset representing a new domain (D_2) that the model must learn. Additionally, two test maneuvers from D_1 and D_2 are included to assess the performance of training on new data, as well as the impact of catastrophic forgetting when introducing the new dataset (D_2) into the model. As shown in Table 4.1,

Algorithm 2: EM-ReSeleCT for Continual Learning

Input: Historical datasets $D_1, \dots, D_k$, and the newly collected dataset in new domain D_{k+1} , where each D_i represents the i^{th} time series dataset, containing sensor measurements for inputs and corresponding target outputs.

Output: Trained model on new data domain, Representative set D^* of historical dataset D

```
1 Pre-train the model on the historical dataset  $D$ .
2 for  $D_j$  in  $\{D_1, D_2, \dots, D_k\}$  /* where  $D_j$  (Eq. (4.3) consists of sensor
   measurements  $\mathbf{x}_j$  and  $\mathbf{y}_j$  as in eq (1),(2) */
3 do
4   for  $\mathcal{Y}_i$  in  $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \dots, \mathcal{Y}_q\}$  /* where  $\mathcal{Y}_i$  is the time series corresponding
   to the  $i$ th output dimension */
5   do
6     Apply change point detection (NOT method) to the sensor measurements of
       dataset  $\mathcal{Y}_i$ .
7     Within a fixed distance  $d_{multi}$  for each captured point  $y_{ig}^j$  in  $i$ , calculate the GLR
       difference based on Eq. (4.18).
8     if  $E < threshold$  then
9       | remove the captured point  $y_{ig}^j$ .
10    else
11      | keep the captured point  $y_{ig}^j$ .
12     $Y^{*M} \leftarrow Y_i^*$ : Store the captured points from the  $i$ th dimension along with their
      associated outputs in other dimensions to memory.
13     $D^* \leftarrow D_j^*$ : Return and store the final representative set (all input-output pairs as
      landmarks) for dataset  $D_j$  as  $D_j^*$ .
14 Return representative set from all datasets  $D^* = \{D_1^*, D_2^*, \dots, D_k^*\}$ .
15 Receive a dataset in the new domain  $D_{k+1}$ .
16 In each training epoch, update the model parameters using the projected gradient:
17 if  $\|g_{k+1}\| \geq \|g_k^*\|$  then
18   |  $\tilde{g} = g_{k+1} - \frac{\langle g_{k+1}, g_k^* \rangle}{\|g_k^*\|_2^2} g_k^*$ .
19 else
20   |  $\tilde{g} = g_k^* - \frac{\langle g_{k+1}, g_k^* \rangle}{\|g_{k+1}\|_2^2} g_{k+1}$ .
21 Apply steps 3 to 14 when new batch of data arrives.
22 Wait for the next time series dataset.
```

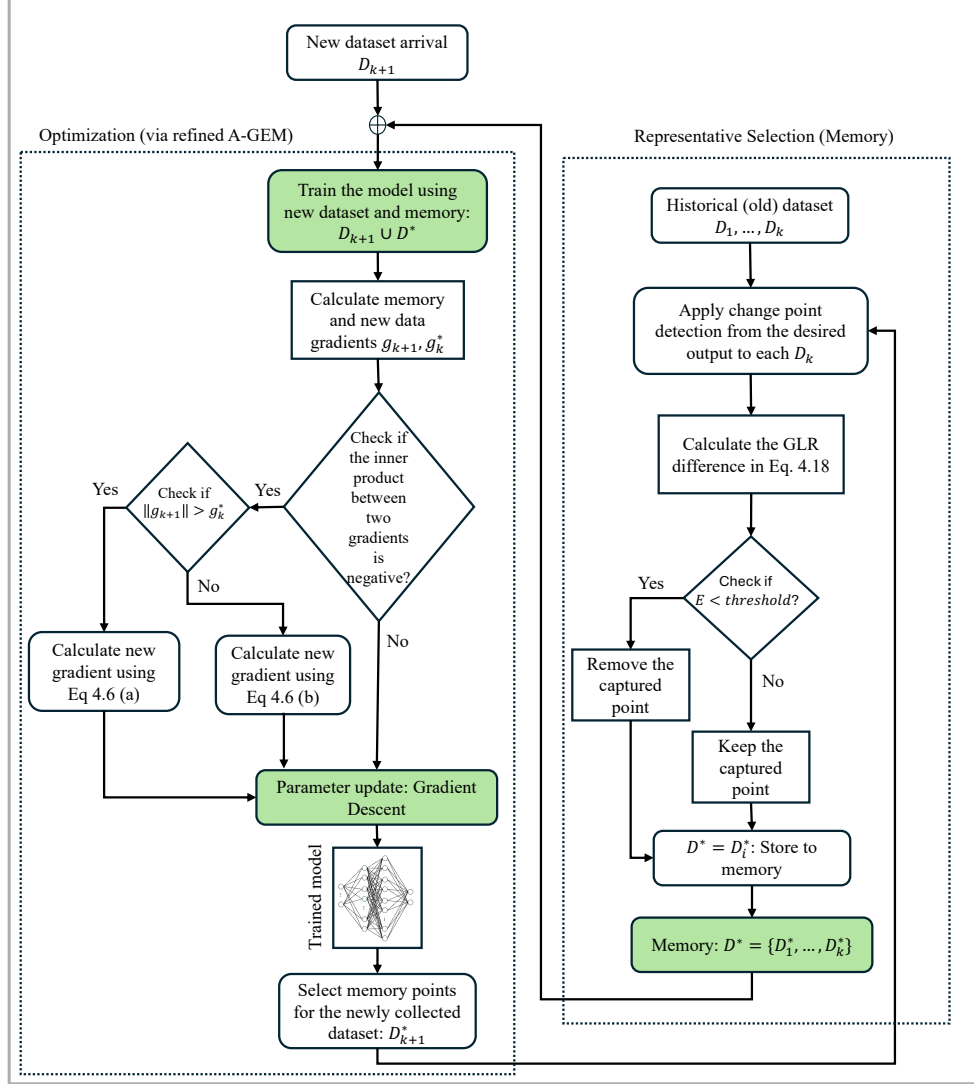


Figure 4.4: Step-by-step procedure of the proposed method (EM-ReSeleCT).

the initial training dataset comprises 10 maneuvers, while the new dataset consists of one maneuver under different road conditions. Measurements were recorded at a frequency of 200HZ, and donwsampled to 100 HZ, capturing various sensor data such as yaw rate, wheel speeds, etc. This section is divided into five subsections. The first subsection details the procedure for the initial training of the neural network model, including hyperparameter settings, input feature selection, and training on the initial dataset. In Subsection 4.5.2, the continual learning scenario is described, where various methods for continually learning the new dataset (D_2) in a new domain, while retaining knowledge from the initial domain, are implemented and evaluated. Subsection 4.5.3 presents a comparison of the performance of the proposed and A-GEM algorithm on memory loss minimization. Additionally, in Subsections 4.5.4 and 4.5.5, an uncertainty quantification of the neural network model and a comprehensive feature selection and sensitivity analysis is performed.

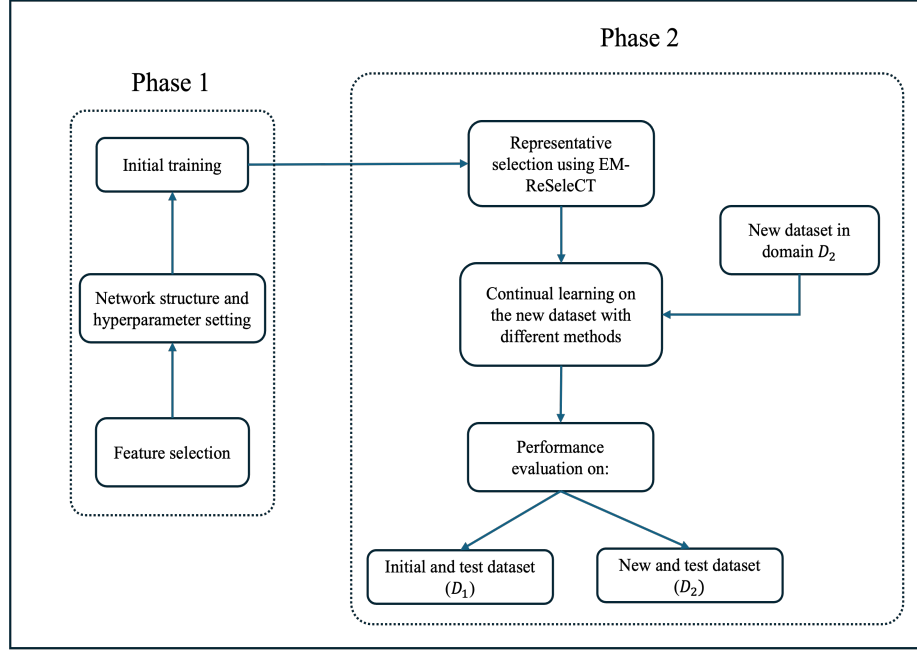


Figure 4.5: The flowchart of evaluation process of different methods in continual learning scenarios.

Table 4.1: Dataset used for evaluation of different methods to estimate vehicle velocity in a continual learning scenario.

Data type	Weather condition	# of maneuvers	# of data
Training: D_1	Sunny(high μ)	10	12000
New Training: D_2	Rainy(low μ)	1	1200
Testing: D_1	Sunny(high μ)	1	1200
Testing: D_2	Rainy(low μ)	1	1200

4.5.1 Phase 1: Initial Training

In the initial step of the evaluation, the model parameters are trained using the initial dataset. This step is common across all continual learning techniques, and the same trained model is used in the subsequent sections when new maneuvers are added. The focus is on thoroughly training the model with the initial dataset to analyze the impact of forgetting on continual learning. As shown in Table 4.1, the initial dataset consists of 10 maneuvers. The objective is to estimate both longitudinal and lateral velocities of the vehicle using input measurements and predicted outputs from previous steps (autoregressive structure) over the period of maneuver time.

Feature Selection

To select the relevant inputs for estimating longitudinal (V_x) and lateral (V_y) velocities, in addition to the last predicted states of both V_x and V_y using the neural network model, a comprehensive analysis has been conducted in Subsection 4.5.5. The resulting features are four wheel speeds, yaw rate, steering angle, and longitudinal and lateral accelerations.

Neural Network Model and Hyperparameter Set

Transformer models, well-suited for handling sequential and time series data, and originally introduced in [62] are used. In the proposed implementation, the encoder inputs consist of sensor measurements and relevant signals, structured as historical values using a time delay embedding, while the decoder inputs are the previously predicted outputs at each time step. This design leverages the historical sensor information to inform the model and uses prior output predictions as additional context for future state estimation. Fig. 4.6 illustrates the complete structure of the proposed Transformer model. Each input series is embedded using a time delay method, where the input at time i includes its current

value alongside the preceding $d - 1$ values. This embedding ensures the Transformer model captures temporal dependencies effectively. Below, the detailed structure and process of the proposed neural network is outlined:

Input Preprocessing: The input signals provided to the encoder include sensor measurements. Each input instance is represented as a tensor of dimensions $p \times d$, with p denoting the number of input features and d representing the time delay window. These inputs are normalized by min-max normalization.

Encoder Upsampling and Positional Embedding: Post-normalization, the encoder inputs are transformed into a higher-dimensional feature space via linear layer upsampling, enhancing the representation capabilities of the neural network. Upsampling the dimension by a factor of four yields optimal results. After upsampling from dimension p to d_{dim} , positional embeddings based on the sine-cosine formulation from the original Transformer architecture are added.

Encoder Attention Blocks: The encoder consists of two sequential self-attention blocks. The first attention block directly follows the positional embeddings, while the second is applied after temporal downsampling via dilated convolutions. These blocks effectively capture both spatial feature-level relationships and temporal dependencies within the input sequences.

Encoder Downsampling: After each attention block, a dilated convolution layer (conv1d) is used, which reduces the temporal dimension by two, to compress the sequence representation, as suggested in [74, 75]. This temporal downsampling contributes significantly to computational efficiency and temporal pattern extraction.

Decoder Structure: The decoder inputs consist of previously predicted output values at each step, allowing the Transformer to condition future predictions based on past estimated states. Similar to the encoder, the decoder applies positional embeddings followed by self-attention mechanisms, specifically used to extract temporal dependencies and relationships in the predicted outputs.

Feedforward Layers: After passing through the attention and convolutional operations, the resulting tensor, with shape $d \times \frac{d}{4}$, is passed by multiple feedforward layers. These layers map the learned features to the desired output dimensions. Specifically, the final output dimensions include estimations for longitudinal and lateral velocities. The final layer is linear, mapping from dimension l_{n-1} (hidden nodes of the preceding layer) to the $q = 2$ output states.

Let $p = 8$ represent the dimensionality of each input time step and d the time delay for the entire sequence. A time delay of 0.5 seconds (50 measurements) is considered for this

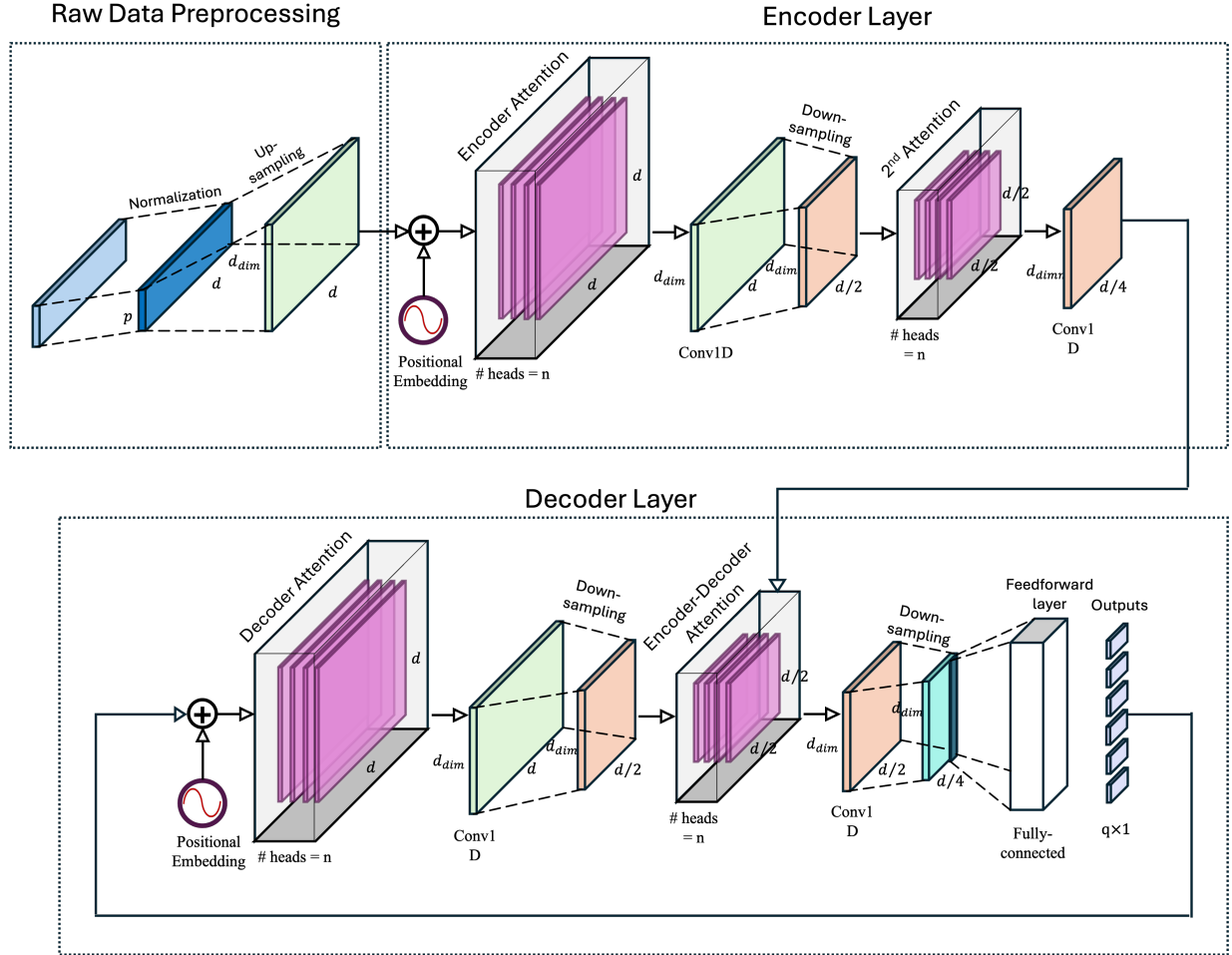


Figure 4.6: Structure of the Transformer model used in this experimental study.

experimental analysis. The input to the Decoder block consists of the last predicted output states of the model, allowing it to use previously predicted states alongside the output of the Encoder layer in the attention head. The decoder inputs are in a q -dimensional space, $q = 2$ as the goal is to estimate V_x and V_y , with a time delay, which is set to match the time delay, d , of the encoder inputs. Finally, the output of this block is processed through a feedforward layer with ReLU activation to introduce nonlinearities, followed by a linear layer that maps the preceding outputs to the final output layer, which has a dimension corresponding to the number of predicted outputs ($q = 2$). The specifications of the hyperparameters for the Transformer model and the proposed continual learning method are outlined in Table 4.2.

Table 4.2: Hyperparameter setting of the Transformer model for estimation of V_x and V_y .

Label	Description	Value
l_e	Encoder hidden layers	2
l_d	Decoder hidden layers	2
A_t	Attention heads	2
h	Feedforward hidden units	500
d	Window size (time delay embedding)	50
m_i	Biased first moment	0.9
v_i	Biased second moment	0.95
α	Learning rate for initial training	0.01
α'	Learning rate for continual learning	0.001
d_{multi}	Distance for the multi-output memory selection	0.1
$threshold$	Threshold value for multi-output memory selection	0.3

Training and estimation results on initial dataset

During initial training, the model is stopped once the MSE loss falls below a predefined threshold. As previously mentioned, an autoregressive Transformer model is used to estimate each successive step of the maneuver. In this setup, the model feeds its prior output estimation into the input decoder rather than the actual output values, and estimation continues until the maneuver completes (typically 12 seconds in the experiments). For the test data, as noted, experiments were conducted within the same domain as the initial dataset to later assess the effect of catastrophic forgetting upon domain expansion. Table 4.3 presents the training time and estimation results on both the initial and test

datasets, while Figure 4.7 illustrates the model’s estimation results on the test data within domain D_1 following initial training.

Table 4.3: Estimation results of V_x and V_y on initial and test dataset after initial training (D_1).

Dataset	Maneuver	Training time (s)	MAE (km/h)	
Initial	10 maneuvers in D_1	247	V_x	0.31
			V_y	0.12
Test	driving at high steering	—	V_x	0.37
			V_y	0.17

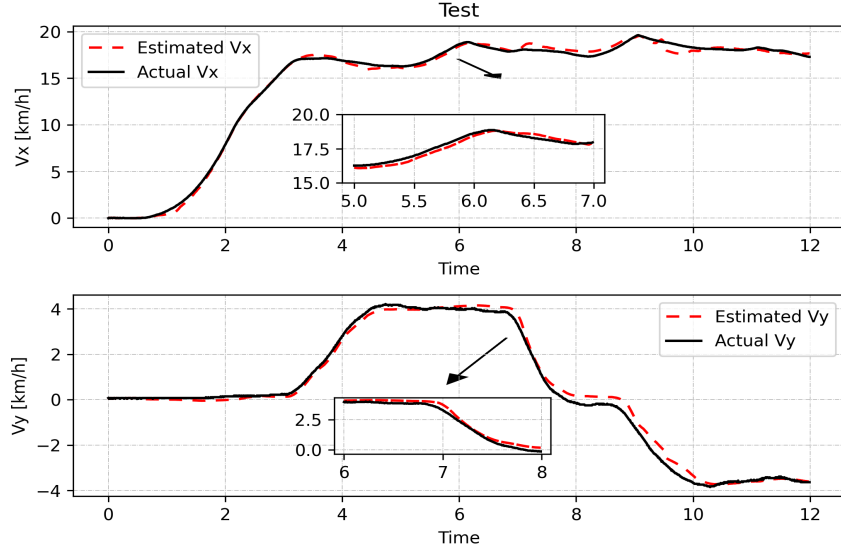


Figure 4.7: Estimation of V_x and V_y on the test data (domain D_1), after the initial training.

As shown in Table 4.3 and Figure 4.7, the model has an acceptable performance in terms of estimation error, as the maneuvers were performed in the same data domain as the initial dataset, there is no need to adapt the model to the test data.

4.5.2 Phase 2: Continual Learning

In this section, a new dataset, collected in a different domain (D_2) compared to the initial and test datasets (D_1) is introduced to the model. Figure 4.8 presents the model’s

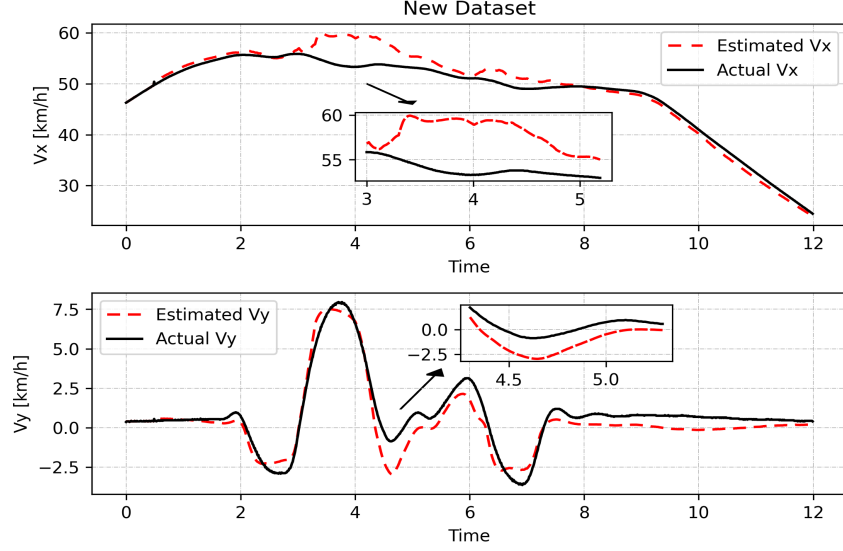


Figure 4.8: Estimation of V_x and V_y on the new data (domain D_2), prior to continual learning.

estimation results on the new dataset before incorporating the new dataset into the model (prior to continual training). As shown in the figure, the model performs poorly on the new dataset. This result is expected, as the maneuver was conducted under significantly different road conditions (heavy rain), requiring the model to adapt to the new dataset. To incorporate the new dataset information into the model, the proposed EM-ReSeleCT alongside other methods are evaluated for comparison, as follows:

Batch: Retrains all neural network parameters using both previously collected and newly acquired data, serving as a baseline or upper bound. This approach, referred to as Batch or Joint training mode.

None: Trains the model solely on the new maneuver, disregarding previous dataset, which represents a lower bound.

A-GEM [7]: Implements the original A-GEM method, in which memory is selected by randomly sampling from the previous task.

SI [29]: Adds a regularization term to the cost function to preserve significant weights of the neural network, based on the sensitivity of each parameter.

A specified number of points is considered as memory points for A-GEM ($m_1 = 140$), using the same quantity for EM-ReSeleCT. However, the key difference in EM-ReSeleCT is

Table 4.4: MAE and training time of V_x and V_y estimation on the new maneuver (D_2) following model training.

Maneuver	Method	Training time	MAE	
			V_x	V_y
D_2	Batch	65.2	0.81	0.17
	None	7.1	0.70	0.11
	SI	8.0	0.74	0.14
	A-GEM	10.5	0.84	0.14
	EM-ReSeleCT	10.9	0.71	0.15

the selection of a specific representative set rather than random selection. This multivariate analysis within EM-ReSeleCT’s structure significantly reduces the number of memory points required. In the proposed approach, after processing historical data with EM-ReSeleCT, the total number of memory points is reduced to $m_1 = 97$. Table 4.4 presents the training time and Mean Absolute Error (MAE) of each continual learning strategy on the new dataset after introducing the domain into the model. As shown, all strategies demonstrate strong performance in estimation error for the new maneuver. In terms of training time, the Batch mode requires substantial time due to its use of all historical data combined with the new data during training. Interestingly, despite the additional computations for memory selection in EM-ReSeleCT and A-GEM, they exhibit comparable training times. This efficiency is attributed to the significantly reduced final memory set in EM-ReSeleCT compared to A-GEM or random selection strategies. SI and None modes have the same training time, as neither retains memory points in the continual learning phase. Additionally, Figure 4.9 shows the estimation results of different methods on the test dataset in D_2 , after introducing the new dataset is D_2 to the model. This is a test dataset that were performed in the wet road, the same as the new dataset. As shown, all three methods perform well on the new dataset, as the new data is introduced to the model and the model was trained on the new dataset using three different methods. A critical aspect of continual learning techniques is their capacity to preserve previously learned knowledge across domains encountered so far. Table 4.5 and Figures 4.10 and 4.11 display the estimation results on the initial dataset and test data (in D_1 and D_2) using different continual learning strategies, following the introduction of a new domain (in D_2) to the model. As shown, the proposed method (EM-ReSeleCT) achieves an estimation error comparable to that of the Batch mode, which serves as the baseline. EM-ReSeleCT significantly outperforms other popular CL strategies, such as A-GEM and SI. This superior performance over A-GEM is attributed to EM-ReSeleCT’s capability to select informative memory points

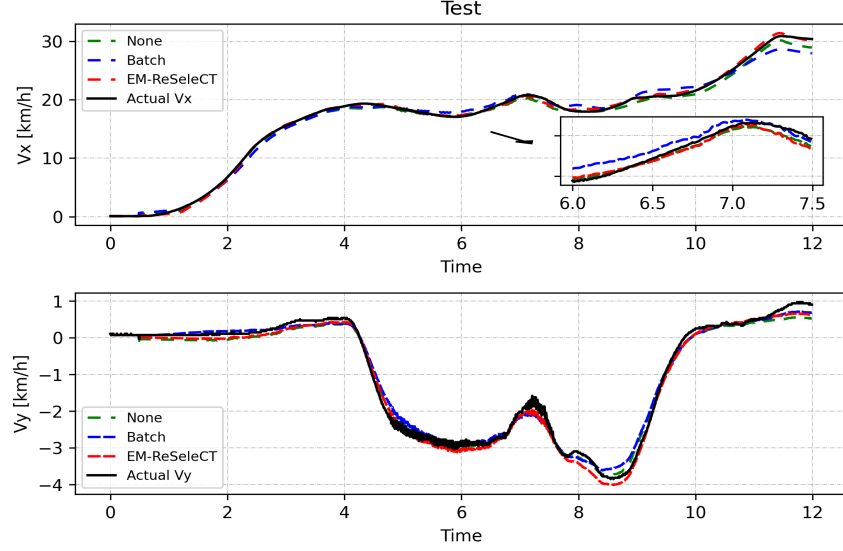


Figure 4.9: V_x and V_y estimations on the test maneuver in domain D_2 , after incorporating the new dataset (in D_2).

rather than relying on random selection, coupled with optimization modifications that enable positive backward transfer, enhancing learning on previously encountered data. SI, on the other hand, shows poor performance, as weight regularization methods have consistently demonstrated limited accuracy in these regression problems. Additionally, as shown in Table 4.5, the all methods have acceptable performance in estimation of both longitudinal and lateral velocities on test 2. Figure 4.10 presents the estimation results of EM-ReSelecT in comparison to the Batch and None modes. As illustrated, for both longitudinal (V_x) and lateral (V_y) velocities, the proposed method demonstrates promising performance, closely aligning with the Batch mode but with significantly lower computational time. In specific regions, EM-ReSelecT effectively preserves information, whereas in the None mode—where prior information is disregarded—the model performs poorly. Figure 4.11 further compares the estimation results of EM-ReSelecT with A-GEM and SI. As shown, EM-ReSelecT exhibits superior performance, while both A-GEM and SI struggle to retain historical information (in D_1). Over time, this results in divergence, causing estimation accuracy for V_x and V_y to deteriorate substantially.

Table 4.5: MAE of V_x and V_y estimation on training and testing maneuvers (in D_1 and D_2) after introducing the new maneuver (D_2).

Method	Initial Dataset		Test (D_1)		Test (D_2)	
	V_x	V_y	V_x	V_y	V_x	V_y
Batch	0.33	0.17	0.36	0.15	0.41	0.19
None	4.21	1.69	3.09	1.00	0.36	0.12
SI	1.40	0.95	1.56	0.44	0.36	0.20
A-GEM	0.79	0.35	1.32	0.31	0.38	0.20
EM-ReSeleCT	0.47	0.25	0.41	0.23	0.38	0.19

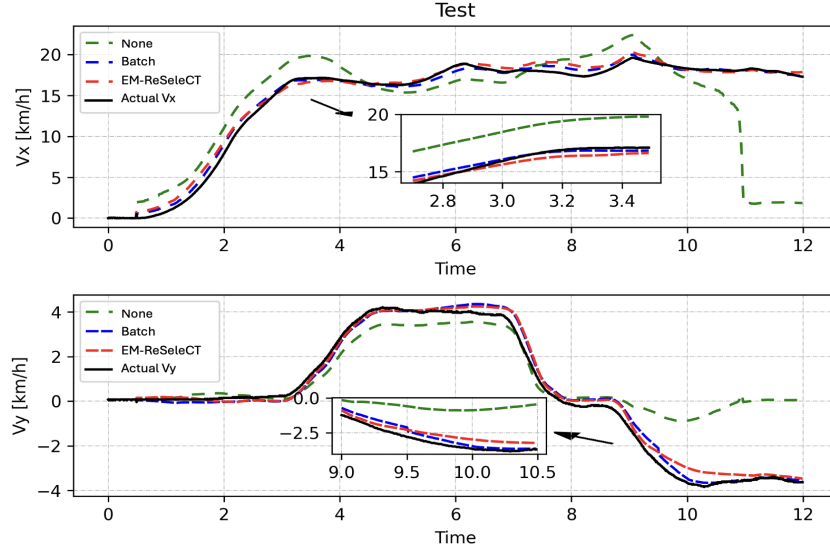


Figure 4.10: V_x and V_y estimations on the test maneuvers (domain D_1), after incorporating the new dataset (in D_2).

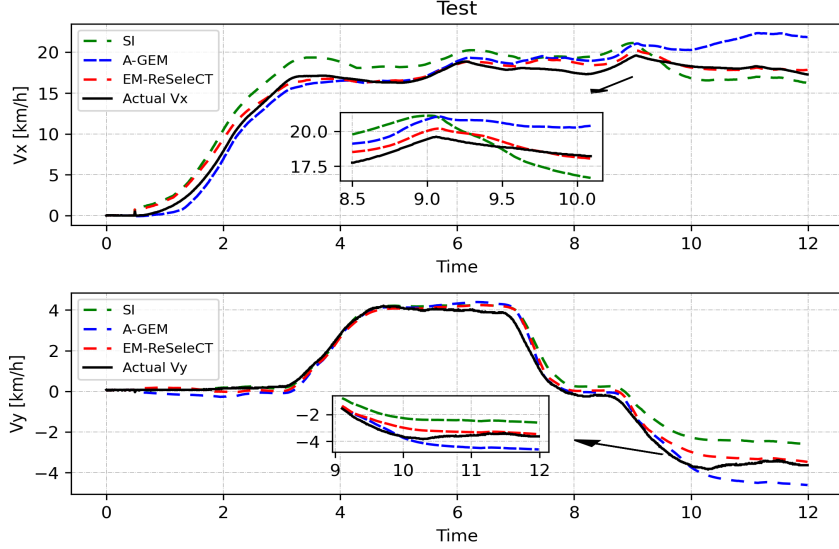


Figure 4.11: Evaluation of the proposed method against other continual learning (CL) strategies for estimating V_x and V_y on test data after incorporating the new dataset (D_2) into the model.

4.5.3 EM-ReSeleCT v.s. A-GEM: Performance on memory loss

The proposed multi-output continual learning approach, EM-ReSeleCT, enhances A-GEM’s optimization algorithm to better preserve historical data, as outlined in Eq. (4.6). Figure 4.12 illustrates the loss on memory points across training epochs in a continual learning scenario. The proposed algorithm not only prevents an increase in memory point loss but also reduces it over certain epochs, demonstrating improved retention of historical information. In addition to the modified optimization, EM-ReSeleCT selects key information by identifying informative data points rather than using A-GEM’s random selection, which further mitigates catastrophic forgetting across all historical data.

4.5.4 Uncertainty Quantification with Conformal Prediction

Validating the performance of models with uncertainty analysis is crucial for assessing their reliability. Conformal prediction (CP) is a statistical framework that provides calibrated prediction intervals for model outputs, ensuring a specified level of confidence. In this work, CP is used to quantify the uncertainty of the proposed model on both the old test dataset

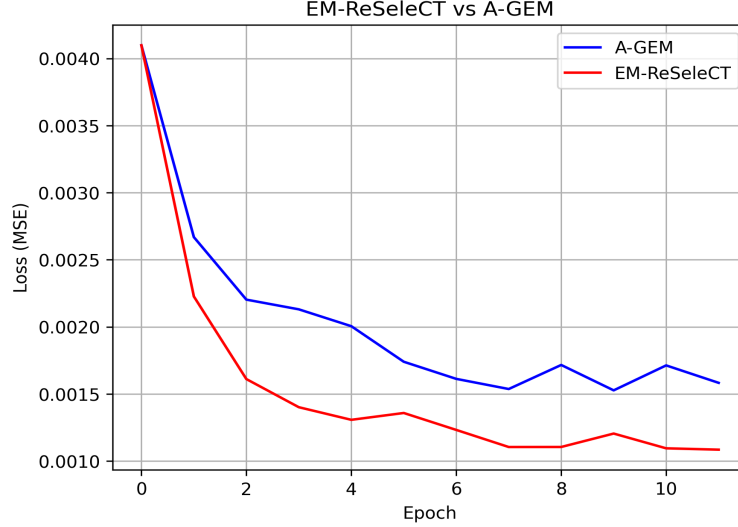


Figure 4.12: Tracking the loss error over each training epoch in continual learning with EM-ReSeleCT and A-GEM.

and the new dataset. Specifically, CP is used to evaluate the uncertainty and coverage rate of the proposed model under different numbers of memory points. It is important to note that split conformal prediction (Vainla CP) as described in [76] is used, which assumes a fixed set of non-conformity scores and a non-exchangeable pair of inputs and outputs. We argue that this assumption holds in the context of this thesis, as the calibration set, after introducing the new maneuver, incorporates both the old and new datasets, and there is no distribution shift between the calibration and test set. Given a prediction model $f(x)$, conformal prediction constructs prediction intervals based on the residuals from a calibration set. Let $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{calib}}$, and $\mathcal{D}_{\text{test}}$ represent the training, calibration, and test datasets, respectively. For each calibration point $(x_i, y_i) \in \mathcal{D}_{\text{calib}}$, the residual is calculated as:

$$r_i = y_i - \hat{f}(x_i), \quad i = 1, \dots, n. \quad (4.19)$$

The quantile of the residuals at level $1 - \alpha$ is computed as:

$$q_{1-\alpha} = \text{Quantile}_{1-\alpha}\{r_1, r_2, \dots, r_n\}. \quad (4.20)$$

For a new test input x_{test} , the prediction interval is then given by:

$$\text{PI}(x_{\text{test}}) = \left[\hat{f}(x_{\text{test}}) - q_{1-\alpha}, \hat{f}(x_{\text{test}}) + q_{1-\alpha} \right], \quad (4.21)$$

where $\hat{f}(x_{\text{test}})$ is the estimated output for input x_{test} , and $q_{1-\alpha}$ is the quantile calculated in Eq 4.20. This interval is guaranteed to cover the true target value y_{test} with probability at least $1 - \alpha$, under the assumption that the calibration set is exchangeable with the test data. In this study, the conformal prediction is applied to estimate the prediction intervals for the outputs V_x and V_y . Separate quantiles were computed for each output dimension, providing customized uncertainty bounds for multi-output predictions. To ensure reliable uncertainty quantification, the conformal prediction model is recalibrated after introducing the new dataset. The calibration data includes samples from both the old and new datasets to ensure comprehensive coverage. After determining the quantile q_α for each model, the uncertainty quantification is applied to both the test data from the old data space and the new data space. Figure 4.13 (a) and (b), show the coverage rate and interval width for the estimation of V_x with different numbers of memory points selected for the proposed model on the test dataset from the old data space. Figure 4.13 (c) and (d), illustrate the coverage rate and interval width for the estimation of V_y with different numbers of memory points selected for the proposed model. As illustrated in the figures, the interval size and uncertainty level increase significantly as the number of memory points decreases. Furthermore, the figures demonstrate that after increasing the number of memory points beyond a certain threshold, the interval size remains unchanged. This suggests that storing additional memory points beyond this threshold does not contribute to further efficiency gains. The reason is that the memory points selected by the automatic approach discussed in the previous section act as representatives of historical data. Over-selecting leads to retaining redundant, non-informative samples that neither enhance information preservation nor effectively mitigate catastrophic forgetting.

4.5.5 On the input sensitivity analysis and feature selection

As described in the previous section, a comprehensive feature selection strategy is done to identify the most relevant input signals for the encoder layer. For this purpose, a sensitivity analysis was conducted to determine which features exert the greatest influence on the neural network model, following the approach suggested in [77]. To quantify the sensitivity of the outputs to each input, the fully trained Transformer model was evaluated on a dataset containing all potential input signals, namely, four wheel speeds ($w_{fl}, w_{fr}, w_{rl}, w_{rr}$), steering angle (δ), yaw rate (ψ), longitudinal and lateral accelerations (ax and ay), angular acceleration (AA_y), yaw acceleration (AA_z), roll rate (AV_x), and vertical acceleration (Az). The sensitivity of an output with respect to a particular input is defined as the change in the output resulting from a small perturbation in the input signal:

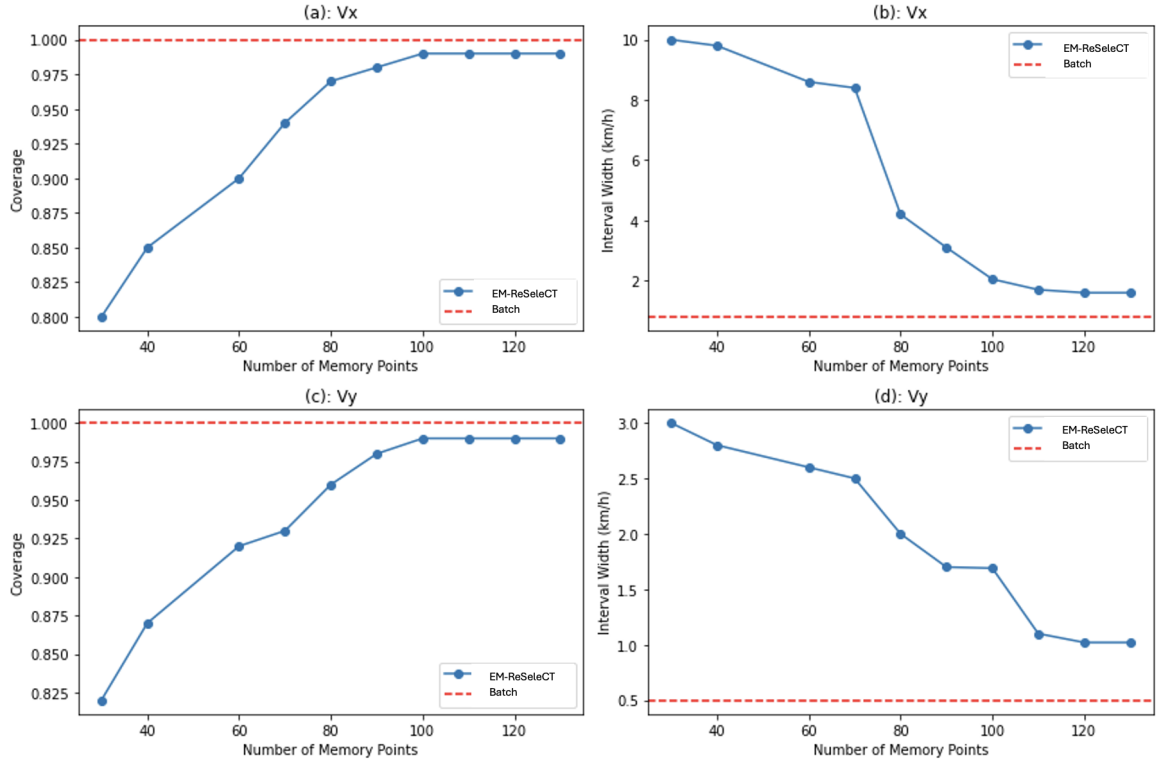


Figure 4.13: Uncertainty analysis of different models by number of selected memory points, on the old (test) dataset. a and b: coverage rate and interval width vs the number of memory points on V_x , respectively. c and d: coverage rate and interval width vs the number of memory points on V_y , respectively.

$$S_{i,j} = \frac{\partial y_j}{\partial x_i} \quad (4.22)$$

where $S_{i,j}$ denotes the sensitivity of the j th output with respect to the i th input. These sensitivities are computed using the gradients of each output with respect to each input signal. Thus, the sensitivity score for each output/input pair can be obtained. Figure 4.14 illustrates the calculated sensitivity scores for each output with respect to each input, and Table 4.6 presents the corresponding values. For clarity, the values were normalized to a maximum of one. As shown, for V_x , the top signals for V_x , exhibit the highest sensitivity scores, indicating that small changes in these signals lead to substantial changes in the corresponding output. Similarly, for V_y , the signals top signals for V_y have the most significant impact. This analysis suggests that the first eight signals are the most influential, justifying the removal of the others. To relate this sensitivity analysis to state

Table 4.6: Sensitivity scores for each input signal with respect to V_x and V_y .

Input Signal	V_x	V_y
w_{fl}	0.0803	0.0612
w_{fr}	0.0713	0.0646
w_{rl}	0.2335	0.0683
w_{rr}	0.3180	0.0995
ψ	0.0437	0.0838
δ	0.0523	0.1534
a_x	0.0835	0.0790
a_y	0.0507	0.2482
AA_z	0.0057	0.0276
AV_x	0.0227	0.0432
A_z	0.0083	0.0408
AA_y	0.0302	0.0305

estimation accuracy, Table 4.7 reports the estimation error for V_x and V_y after sequentially removing the least sensitive signals (starting from the end of the table). The estimation error is computed as:

$$\text{MAE} = \frac{1}{N} \sum_{k=1}^N |y_k - \hat{y}_k| \quad (4.23)$$

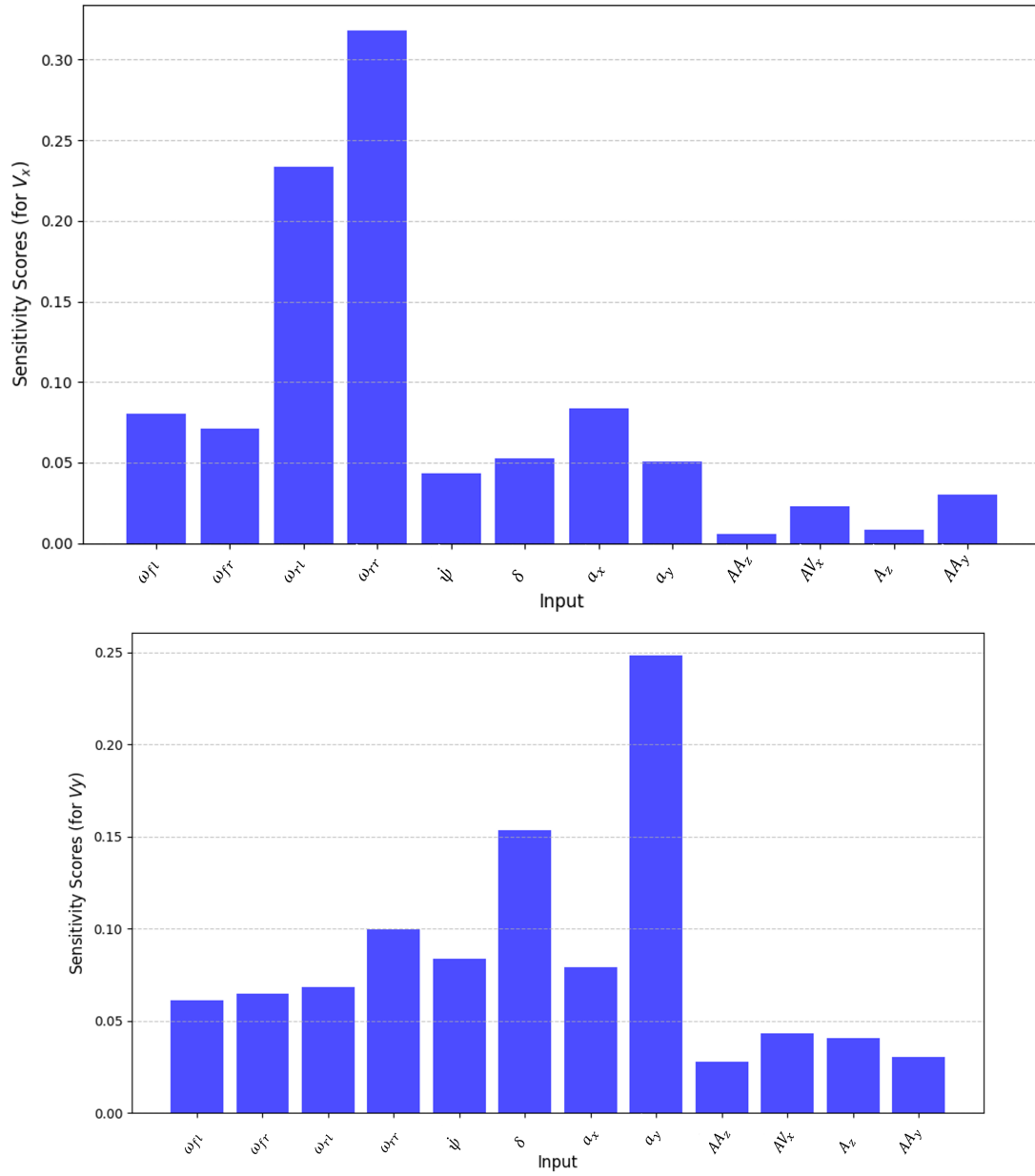


Figure 4.14: Sensitivity of the outputs to the change in each input signal.

where y_k and $\hat{y}_k$ denote the true and estimated values at sample k , respectively, and N is the total number of samples. As shown in Table 4.7, after removing the fifth signal, the estimation error on the test data in the initial dataset domain (D_1) increases significantly, indicating that while the first four signals (the four least sensitivity scores) are less relevant, the fifth signal plays an important role in accurate state estimation. It is worth noting that, after each removal, a new model was trained to estimate the exact same test.

Table 4.7: Estimation error (MAE) for V_x and V_y after sequentially removing the least sensitive input signals.

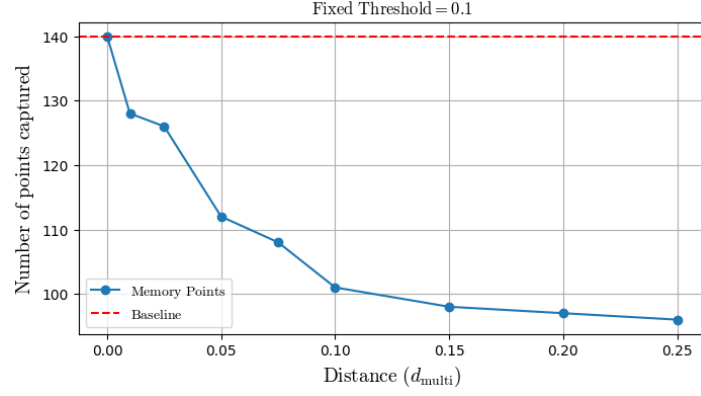
Number of Signals Removed	V_x	V_y
0 (all inputs)	0.32	0.14
1	0.32	0.14
2	0.32	0.15
3	0.35	0.17
4	0.37	0.17
5	0.49	0.27

Another sensitivity analysis conducted in this chapter focuses on the two key parameters used in the EM-ReSeleCT method for multi-output representative selection: the threshold value and the distance parameter (d_{multi}). In general, increasing either the threshold or the distance results in fewer selected points, though the relationship is not strictly linear. To evaluate the impact of each parameter on the number of selected points, we applied the analysis to the historical (initial) dataset used throughout this chapter. Figure 4.15 illustrates how varying each parameter, while holding the other constant, affects the number of selected memory points. It should be noted that the baseline corresponds to applying the proposed representative-selection method independently to each output dimension (V_x and V_y), with no explicit mechanism for eliminating redundant representatives.

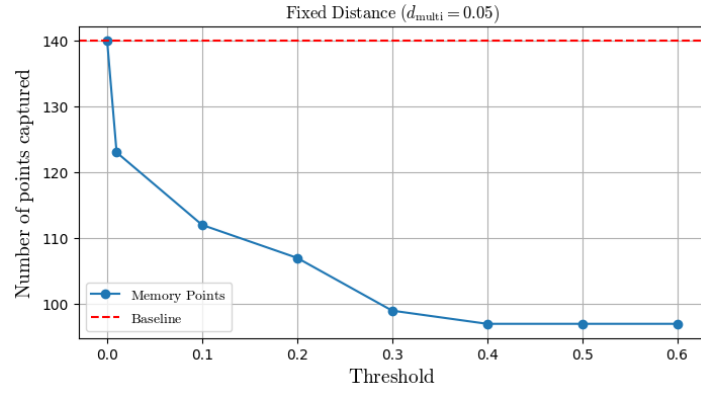
It should be noted that identifying the optimal number of selected points ultimately depends on the estimation error. Therefore, a similar sensitivity analysis, as discussed in Chapter 3, should be performed to relate the number of selected points to the resulting estimation performance. Such analysis ensures that the selected memory configuration achieves an effective balance between model compactness and predictive accuracy.

4.6 Summary

In this Chapter, EM-ReSeleCT, an efficient multivariate representative selection and optimization strategy for continual learning in time series tasks was introduced. A feature selection method using sensitivity analysis of different vehicle signals was conducted to find the most relatable features for the neural network inputs. Experimental analysis on multivariate output estimation using Transformer models, applied to the electric Equinox vehicle, demonstrates the superior performance of EM-ReSeleCT in both training time and estimation accuracy compared to other state-of-the-art continual learning algorithms. Specifically, EM-ReSeleCT achieves estimation errors close to the Batch mode while significantly reducing computational time. The method effectively minimizes the number of memory points by identifying instances where two selected points across outputs represent the same information but are in close proximity.



(a)



(b)

Figure 4.15: Sensitivity of memory point selection to threshold and distance parameters in EM-ReSeleCT; (a): Under fixed threshold. (b): Under fixed distance.

Chapter 5

Developing an adaptive Real-time Hybrid Vehicle State Estimation

5.1 Introduction

So far, the proposed continual learning methods have assumed known task boundaries, where changes in the input and output distributions were explicitly identified in advance. However, in real-world scenarios, task boundaries are often unknown, requiring the detection of distribution shifts in the data. This requires a system to identify these changes and temporarily switches to model-based observers, with continual learning applied once new data becomes available. To address these issues, this chapter presents an adaptive hybrid estimation framework that dynamically switches between a model-based observer and a neural network-based estimator. A distribution-shift detection module identifies when the system encounters a new unknown condition, and triggers the use of the model-based observer to maintain robust state estimation. Simultaneously, a continual learning method incrementally updates the neural network with the new data, ensuring the model adapts to new conditions without reducing the performance on previously seen distributions [78].

5.2 Problem Formulation

Two general estimation strategies are considered: (1) a model-based observer that relies on a mathematical system description, and (2) a neural-network-based estimator that learns directly from data.

The first approach is to use a model-based estimator to estimate the output response, using the input measurements and mathematical model of the system. The following discrete-time nonlinear system with process and measurement noise is considered:

$$\mathbf{x}_i = f_{\text{model}}(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) + \mathbf{w}_i, \quad (5.1)$$

$$\mathbf{y}_i = h'(\mathbf{x}_i) + \mathbf{v}_i. \quad (5.2)$$

A general recursive observer then consists of the prediction and correction steps, followed by the output equation:

$$\hat{\mathbf{x}}_{i|i-1} = \Phi(\hat{\mathbf{x}}_{i-1|i-1}, \mathbf{u}_{i-1}; \theta_{\text{model}}), \quad (5.3)$$

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K'_i \left(\mathbf{y}_i - H'(\hat{\mathbf{x}}_{i|i-1}; \theta_{\text{model}}) \right), \quad (5.4)$$

$$\hat{y}_i = g(\hat{\mathbf{x}}_{i|i}; \theta_{\text{model}}) + \epsilon_{i,\text{model}}. \quad (5.5)$$

Here, $\mathbf{x}_i \in \mathbb{R}^n$ denotes the system state at time step i , and $\mathbf{u}_i \in \mathbb{R}^m$ is the input vector. The output measurement at time i is represented by $\mathbf{y}_i \in \mathbb{R}^p$. The function $f(\cdot)$ describes the nonlinear state transition dynamics, while $h'(\cdot)$ maps the state to the measurement space. The terms $\mathbf{w}_i \sim \mathcal{N}(0, Q_i)$ and $\mathbf{v}_i \sim \mathcal{N}(0, R_i)$ denote the process and measurement noise, respectively. The predicted state estimate prior to measurement update is denoted by $\hat{\mathbf{x}}_{i|i-1}$, and the updated estimate after incorporating the measurement is $\hat{\mathbf{x}}_{i|i}$. The mapping $\Phi(\cdot)$ is the prediction model used in the observer, typically derived from the system dynamics f , and $H'(\cdot)$ is the measurement prediction model corresponding to h' . The gain K'_i represents the adaptive correction gain, such as the Kalman gain in linear or extended Kalman filters. The function $g(\cdot)$ produces the final estimated output, and θ_{model} represents the parameters of the model used in Φ , H' , and g . ϵ_i is the residual modeling error and unmodeled system dynamics.

Alternatively, state estimation can be treated as a regression problem over sequential data, with the estimator model being a neural network. Assume $(\mathbf{x}_i, y_i)$, for $i = 1, 2, \dots$, indicates a set of sequential observations over time, where $y_i \in \mathbb{R}$ is the desired response variable and $\mathbf{x}_i \in \mathbb{R}^p$ is the input or feature vector, and assume the relationship between y and $\mathbf{x}$ for $i = d+1, d+2, \dots$ follows:

$$y_i = f(\mathbf{x}_i, H_i(D)) + \epsilon_i, \quad (5.6)$$

where D is the current dataset under study, f is an unknown function that needs to be estimated, and ϵ_i is the additive noise of observations i , and d represents the temporal dependencies among $(\mathbf{x}_i, y_i)$ s and:

$$H_i(D) = \{(\mathbf{x}_j, y_j) \in D; i - d \leq j < i\}.$$

At each instance i , the goal is to predict the output response y , by estimating the function f on available dataset D :

$$\begin{aligned}\hat{y}_i &= \hat{f}_{\text{NN}}(\mathbf{x}_i; \boldsymbol{\theta}) + \epsilon_i, \\ \ell_i(\boldsymbol{\theta}) &= (y_i - f(\mathbf{x}_i; \boldsymbol{\theta}))^2, \\ R(\boldsymbol{\theta}; D) &= \frac{1}{|D|} \sum_{i=1}^{|D|} \ell_i(\boldsymbol{\theta}).\end{aligned}\tag{5.7}$$

Empirical studies show that when test data lie within the distribution seen during training, neural network or machine learning based estimators often outperform pure model-based observers. However, in the presence of a distribution shift, when the input data deviate from the training distribution, the performance of learning-based estimators deteriorates, and the model-based observer becomes more reliable. This observation raises two critical questions:

- How to detect a distribution shift in real time?
- How to continually update the neural network with new data without forgetting previous information?

Figure 5.1 illustrates the proposed two-stage hybrid estimation procedure. At each time step, upon receiving the new sensor measurement $\mathbf{x}_i$, a distribution-shift detector determines whether $\mathbf{x}_i$ lies within the neural network's training distribution; if no shift is detected, the neural estimator computes

$$\hat{y}_i = \hat{f}_{\text{NN}}(\mathbf{x}_i; \boldsymbol{\theta}) + \epsilon_i,\tag{5.8}$$

otherwise the algorithm falls back to the model-based observer to produce

$$\hat{y}_i = \hat{f}_{\text{model}}(\mathbf{x}_i; \theta_{\text{model}}) + \epsilon_i\tag{5.9}$$

and simultaneously triggers a continual-learning update of the neural network parameters using the newly observed pair $(\mathbf{x}_i, y_i)$, ensuring adaptation to the new information without erasing prior knowledge.

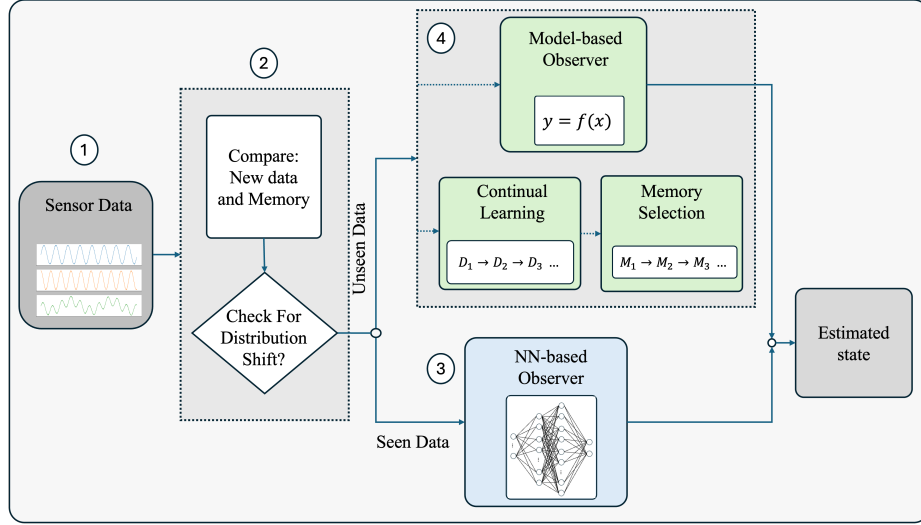


Figure 5.1: The proposed procedure for adaptive state estimation.

5.3 Hybrid State Estimator

To take advantage of both model-based and neural network estimators, it is first determined whether a newly collected input sequence lies within the distribution that the neural network has already seen. If it does, NN-based predictor is used for the estimation task; otherwise, the method falls back to the model-based observer and triggers continual learning. Let $\mathbf{X}_{i,\text{test}} = [\mathbf{x}_{i-d+1}, \dots, \mathbf{x}_i]^T \in \mathbb{R}^{d \times p}$ denote the input window for the current test sample, where d is the window length and p the feature dimension. Let $\mathcal{M} = \{(\mathbf{x}_j^*, y_j^*)\}_{j=1}^m$ be the set of m representative memory points selected via the ReSeleCT algorithm introduced in Chapter 3, and $\mathbf{x}_j^* \in \mathbb{R}^{d \times p}$. Then, the minimum Euclidean distance between the test input and all memory inputs is computed as:

$$d_{\min} = \min_{1 \leq j \leq m} \sum_{k=1}^d \|\mathbf{x}_{i-d+k} - \mathbf{x}_{j,k}^*\|_2. \quad (5.10)$$

A distribution shift is detected if

$$d_{\min} > \tau, \quad (5.11)$$

where τ is a predefined threshold. If $d_{\min} \leq \tau$, we infer that the neural network has seen a similar distribution and proceed with Eq. 5.8. If $d_{\min} > \tau$, the model-based observer is

invoked and use Eq. 5.2 for estimation and simultaneously update the neural network via continual learning on $\mathbf{X}_{i,\text{test}}$ and its true label y_i , while retaining the existing memory $\mathcal{M}$ and adding a representative subset of the new data for future recognition. This two-stage procedure ensures robust estimation and gradual adaptation to novel operating conditions.

5.3.1 Representative Selection and Continual Learning

To maintain a compact yet informative memory of past observations and to detect distribution shifts, the proposed algorithms detailed in the previous chapters, namely ReSeleCT and EM-ReSeleCT are used. Based on eq. 3.11 in Chapter 3, for each retained c , the corresponding d -length input-output segment $(\mathbf{x}_j^*, y_j^*) = \{(\mathbf{x}_{j,c}, y_{j,c})\}_{j=c-\frac{d}{2}}^{c+\frac{d}{2}}$ is added to the memory set

$$\mathcal{M} = \{(\mathbf{x}_j^*, y_j^*)\}_{j=1}^m.$$

By construction, $\mathcal{M}$ contains the most informative and non-redundant samples needed both for retraining and for shift detection. During testing, when the minimum Euclidean distance $d_{\min}$ between the current window $\mathbf{X}_{i,\text{test}}$ and the memory inputs exceeds the shift threshold τ , a distribution shift is declared. In that case, the estimator immediately falls back to the model-based observer, and the neural network is trained with a continual learning method. Specifically, the network is retrained on the the new sample $(\mathbf{X}_{i,\text{test}}, y_i)$ and from $\mathcal{M}$, thereby preserving previous information while integrating the new dynamics. If $(\mathbf{X}_{i,\text{test}}, y_i)$ represents a particularly new condition, that is not already well captured by $\mathcal{M}$, it is also added to $\mathcal{M}$. This continuous interplay between robust model-based estimation and data-driven adaptation ensures reliable performance across both seen and previously unseen operating conditions.

Algorithm Description. Algorithm 3 processes each incoming sensor measurement by first computing its minimum Euclidean distance to a stored memory of representative past samples. If the distance is below the threshold τ , the pretrained neural network is used for prediction. Otherwise, the algorithm falls back to the model-based observer for a reliable estimate and simultaneously performs a continual-learning update of the neural network using the new sample combined with a memory points. Finally, the memory is updated via ReSeleCT or EM-ReSeleCT to retain the most informative samples, ensuring both robustness to distribution shifts and adaptability to new operating conditions.

Algorithm 3: An adaptive real-time Hybrid state estimation

Input: $\mathcal{M} = \{(\mathbf{x}_j^*, y_j^*)\}_{j=1}^m$: memory of past representative samples,

Trained neural network $\hat{f}_{\text{NN}}(\cdot; \boldsymbol{\theta})$,

Mathematical model $\hat{f}_{\text{model}}(\cdot; \theta_{\text{model}})$,

Threshold τ

Output: Estimated output sequence $\{\hat{y}_i\}$

```
1 for each time step  $i = 1, 2, \dots$  do
2   Receive new sensor input  $\mathbf{X}_{i,\text{test}} = [\mathbf{x}_{i-d+1}, \dots, \mathbf{x}_i]^T \in \mathbb{R}^{d \times p}$ 
3    $d_{\min} = \min_{1 \leq j \leq m} \sum_{k=1}^d \|\mathbf{x}_{i-d+k} - \mathbf{x}_{j,k}^*\|_2$ 
4   if  $d_{\min} \leq \tau$  then
5      $\hat{y}_i \leftarrow \hat{f}_{\text{NN}}(\mathbf{x}_i; \boldsymbol{\theta})$ 
6   else
7      $\hat{y}_i \leftarrow \hat{f}_{\text{model}}(\mathbf{x}_i; \theta_{\text{model}})$ 
7     // Continual learning update
8     Retrain  $\hat{f}_{\text{NN}}$  on  $\{(\mathbf{x}_i, y_i)\}$  in new domain and random mini-batch from  $\mathcal{M}$ 
8     from old domain;
8     // Memory refresh via ReSelecT
9      $\mathcal{M} \leftarrow \text{ReSelecT}(\mathcal{M} \cup \{(\mathbf{x}_i, y_i)\})$ 
10 return  $\{\hat{y}_i\}$ 
```

5.4 Evaluation and Testing

In this experiment, both model-based and neural network-based observers are used to estimate the longitudinal and lateral velocities. The same vehicle described in Chapters 3 and 4 is used for the experimental tests. The maneuvers are derived from three different road conditions: dry, wet, and icy surfaces. For the model-based estimator, the following observer equations for longitudinal and lateral velocities are used as presented in [70]:

$$\dot{\hat{v}}_x = a_x + \sum_{i=1}^4 K'_i(a_x, \omega) (v_{x,i} - \hat{v}_x) \quad (5.12)$$

$$\dot{\hat{v}}_y = -v_x \dot{\psi} + a_y - K'_{v_y} \left(m a_y - \sum_{i=1}^4 (0, 1) \mathbf{R}(\delta_i) \hat{F}_i \right) \quad (5.13)$$

where a_x and a_y denote the measured longitudinal and lateral accelerations, respectively; v_x is the longitudinal velocity of the vehicle; $\hat{v}_x$ and $\hat{v}_y$ are the estimated longitudinal and lateral velocities; $v_{x,i}$ represents the longitudinal velocity of wheel i , computed from wheel speed measurements; $\dot{\psi}$ is the vehicle yaw rate; m is the vehicle mass; $\hat{F}_i$ is the friction tire force at wheel i , estimated by the tire model; $\mathbf{R}(\delta_i)$ is the rotation matrix corresponding to the steering angle δ_i at wheel i , used to transform the wheel force into the body-fixed frame; $(0, 1)$ is a row vector that extracts the lateral component of the force vector; $K'_i(a_x, \omega)$ denotes an adaptive observer gain for wheel i , which depends on the longitudinal acceleration a_x and the vector $\omega = [\omega_{1=fl}, \omega_{2=fr}, \omega_{3=rl}, \omega_{4=rr}]^T$ of wheel speed measurements; and K'_{v_y} is the observer gain for the lateral velocity estimation. For further details on the state-space representation, the derivation of the observer equations, and the proof of stability, the reader is referred to [70]. Note that the observer used is a three-state space model (longitudinal and lateral velocities along with the yaw rate). The equations above correspond to the separate (modular) observer, while the equations below represent the combined observer:

$$\dot{\hat{v}}_x = \hat{v}_y \dot{\psi} + a_x + \sum_{i=1}^4 K'_i(a_x, \omega) (v_{x,i} - \hat{v}_x) \quad (5.14)$$

$$\dot{\hat{v}}_y = -\hat{v}_x \dot{\psi} + a_y - K'_{v_y} \left(m a_y - \sum_{i=1}^4 (0, 1) \mathbf{R}(\delta_i) \hat{F}_i \right) \quad (5.15)$$

$$\dot{\hat{\psi}} = \frac{1}{J_z} \sum_{i=1}^4 \mathbf{g}_i^T \mathbf{R}(\delta_i) \hat{F}_i + K_r' (\dot{\psi} - \dot{\hat{\psi}}) \quad (5.16)$$

Here, $\mathbf{g}_i$ denotes the geometry vector from the vehicle center of gravity to wheel i , while K_r' is the observer gain associated with the yaw rate correction. Note that, in this Chapter, for the model-based observer, we assumed we have a good estimate of the friction model, and the wheel slips in harsher scenarios, but the model still has uncertainties, requiring to use recursive methods to reduce the uncertainty. For the neural network-based observer, the Transformer model introduced in Chapter 4 is used to estimate both longitudinal and lateral velocities at each time step.

5.4.1 Initial training and estimation results on the dry road

For the initial training of the neural network, experiments conducted under dry road conditions were used. The input signals, as illustrated in Figure 5.2, consist of eight key measurements selected based on the feature selection, as previously discussed in Chapters 3 and 4. These include the four individual wheel speeds, steering angle, yaw rate, and both longitudinal and lateral accelerations. Each of these input signals is provided to the encoder along with their corresponding time delays. Additionally, the previously estimated outputs, namely, longitudinal and lateral velocities, are supplied to the decoder at each time step. This input selection is designed to capture the essential vehicle dynamics required for accurate state estimation.

During the initial learning phase, training is stopped once the MSE loss reaches below a certain threshold. The same neural network architecture is used as was detailed in Chapter 4, where an autoregressive Transformer with an encoder-decoder mechanism was proposed. For testing, experiments were carried out under the same domain conditions as those used for the initial dataset. The hyperparameters used in this study were selected based on the procedure described in Chapter 4. It is worth noting that the final hyperparameters represent the best-performing configuration identified during the tuning process. Figures 5.3 and 5.4 show the estimation results of the trained neural network on a maneuver performed on a dry road for the estimation of longitudinal and lateral velocities, respectively. As expected, since the neural network was trained on data from this operating condition, it accurately estimates both the longitudinal and lateral velocities with minimal error. In this scenario, there is no need to switch to a model-based observer, as the neural network estimator demonstrates sufficiently high performance.

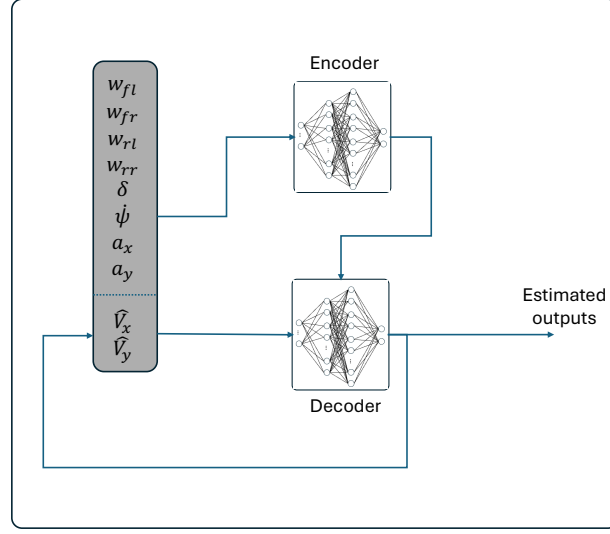


Figure 5.2: Input signals to the encoder and decoder layers at each time step.

5.4.2 Estimation under Distribution Shift: Wet and Icy Roads

After training the model on maneuvers performed on dry road conditions, its ability to detect distribution shifts under changing road surfaces is evaluated. In the first experiment, data collected from maneuvers on a wet road is introduced, where the input distribution differs from the training data. This change is particularly evident in the wheel speed measurements, as some wheels experience slip due to reduced tire-road friction. In such scenarios, the neural network-based estimator may no longer provide reliable results, requiring a switch to the model-based observer while the neural network is updated with new data. Therefore, real-time detection of distribution shifts becomes essential for maintaining accurate state estimation and ensuring robust system performance.

To evaluate the effectiveness of the proposed adaptive hybrid approach, it is compared against both the neural network (NN)-based estimator and the model-based estimator. Figure 5.5 presents the estimation results of all three methods. As illustrated, before second 2, the NN-based estimator outperforms the model-based estimator, since the input data remains within the distribution seen during training and no wheel slip occurs. However, after second 2, when two wheels begin to slip due to reduced friction on the wet road, the model-based estimator has more accurate results. Ideally, the NN-based estimator should be used prior to second 2, and the model-based estimator should be used thereafter. This demonstrates the importance of accurately detecting the distribution shift at second 2, a task that the proposed hybrid estimator accomplishes effectively. Note that for this specific

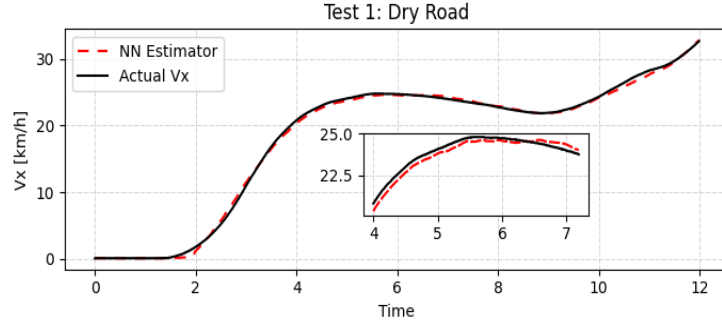


Figure 5.3: V_x estimation on the test maneuvers (domain D_1), after training the model on the initial dataset (dry road) with no distribution shift.

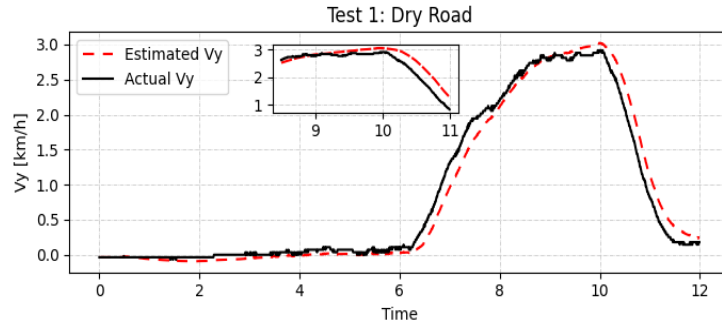


Figure 5.4: V_y estimation on the test maneuvers (domain D_1), after training the model on the initial dataset (dry road), with no distribution shift.

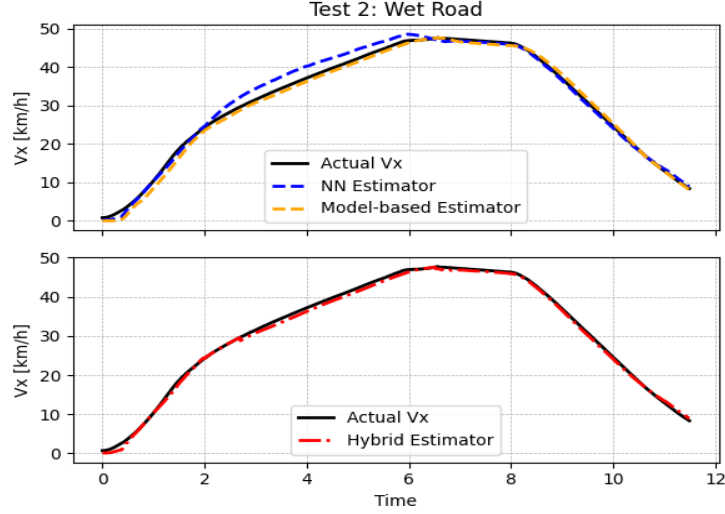


Figure 5.5: V_x estimation on under the distribution shift (wet road), using NN, model, and hybrid-based estimators.

maneuver, only the longitudinal velocity estimation is shown, as the maneuver does not have sufficient lateral velocity.

The third experiment is conducted under icy road conditions, representing a more extreme and harsh scenario. In this case, all wheels experience slip, resulting in a significant change in the data distribution during certain periods of the maneuver. Figures 5.6 and 5.7 present the estimation results on the estimation of longitudinal and lateral velocities. As shown, the hybrid estimator for both the longitudinal velocity in Figure 5.6 and the lateral velocity in Figure 5.7 successfully switches to the model-based observer when a distribution shift is detected. It is important to note that even the model-based observer does not provide ideal estimates under severe wheel slip conditions. However, its performance remains substantially better than the NN-based estimator in the presence of such distribution shifts.

5.4.3 Continual Learning

In this subsection, the continual learning approach proposed in Chapter 3 and 4 is used to identify how it can enhance the performance of the NN-based estimator after detecting a distribution shift. Specifically, the EM-ReSeleCT method is developed, which combines representative selection and the introduced optimization strategy, to continually update the

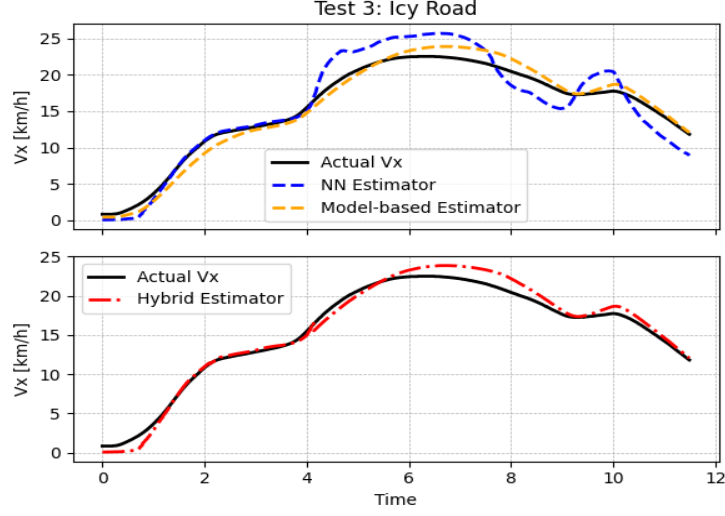


Figure 5.6: V_x estimation on under the distribution sift (icy road), using NN, model, and hybrid-based estimators.

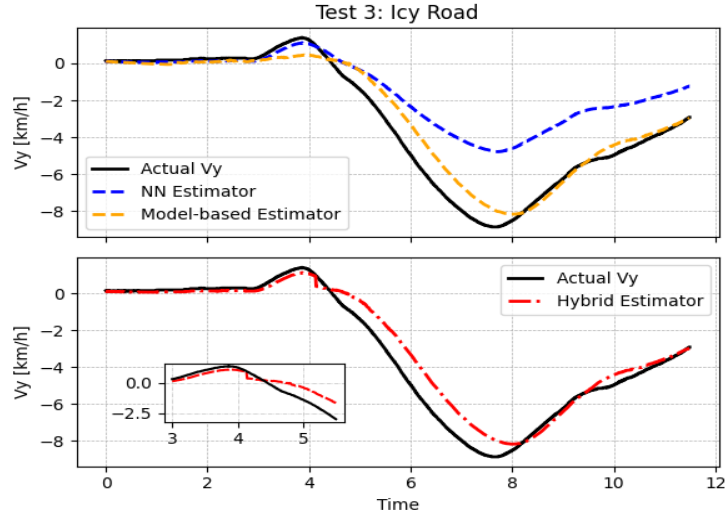


Figure 5.7: V_y estimation on under the distribution sift (icy road), using NN, model, and hybrid-based estimators.

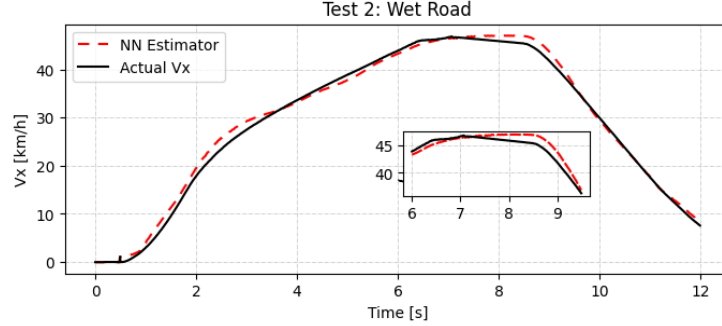


Figure 5.8: V_x estimation on the wet road, after continual learning of the model using the new data distribution.

model with new data. Figures 5.8 and 5.9 illustrate the improved estimation results of the NN-based estimator after continual training on the new dataset. Furthermore, Figure 5.10 highlights the minimal impact of catastrophic forgetting on dry road estimation, confirming that the proposed method effectively mitigates this issue.

5.5 A Comparative Study on the Choice of Neural Network Model for State Estimation

So far, the proposed method has been developed for continual learning scenarios in both univariate and multi-output cases in the context of vehicle state estimation. For the state estimation task itself, different neural network models were used. Specifically, both Recurrent Neural Networks (RNNs), as well as Transformer models were applied to the intended application. It is important to examine the benefits, strengths, and limitations of each neural network architecture in this application. In this regard, a comprehensive comparative study was conducted for the vehicle state estimation task using three different neural network models: a non-autoregressive neural network (NARX), LSTM, and Transformer. For the LSTM model, the same structure described in Chapter 3 was used, with the modification of predicting two output variables. Similarly, for the Transformer model, the same architecture as described in Chapter 4 was adopted. Below is a brief description of the NARX model used in this comparative study:

In NARX, the output at each time step is predicted using both the current input and previous outputs. The network consists of input, hidden, and output layers. In the hidden layers of NARX, multilayer perceptrons with fully connected layers are used. In this study,

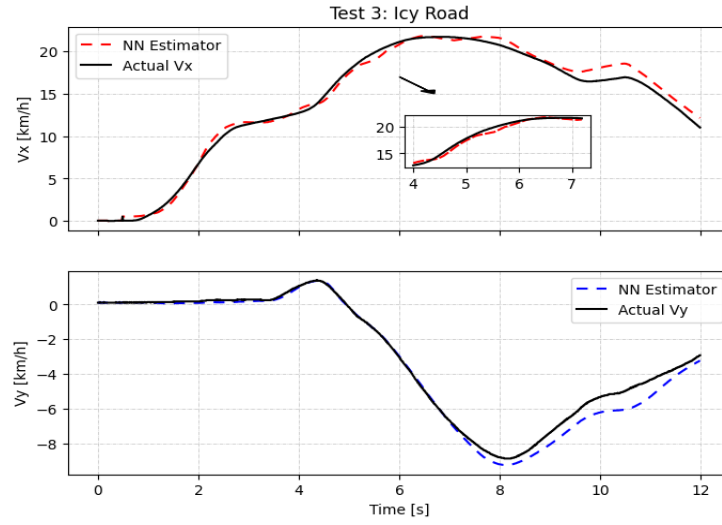


Figure 5.9: V_x and V_y estimation on the icy road, after continual learning of the model using the new data distribution.

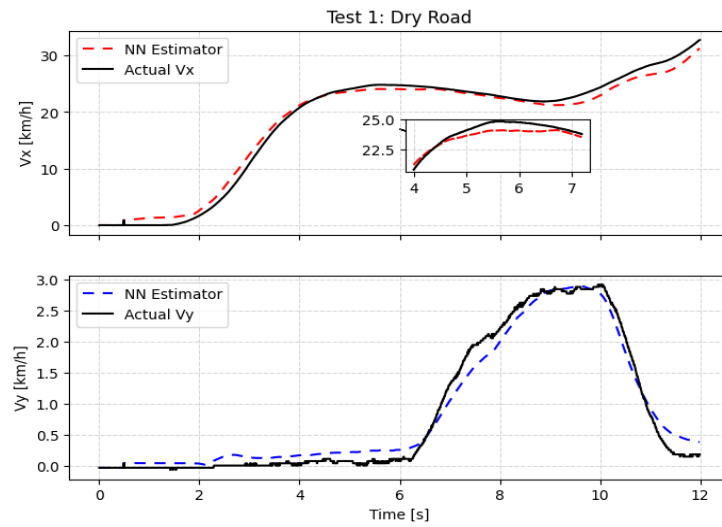


Figure 5.10: V_x and V_y estimation on the dry road (initial dataset), after continual learning of the model using the new data distributions.

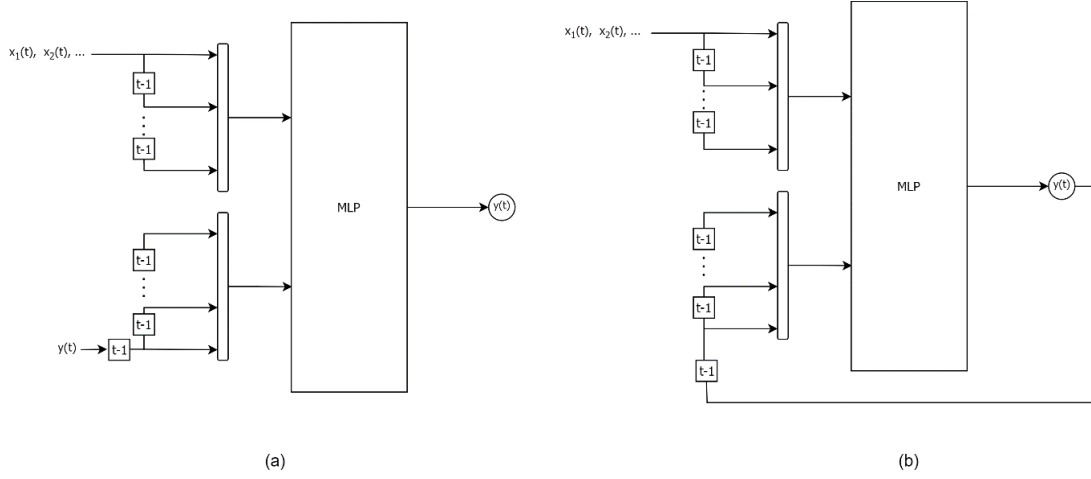


Figure 5.11: NARX Architecture: (a) Series-parallel structure. (b) Parallel structure.

a sigmoid activation function is adopted for each unit in the hidden layers. Two distinct architectures for NARX are commonly used: the series-parallel architecture for training and the parallel architecture for prediction, as shown in Figure 5.11.

In the series-parallel structure, the network is trained using the actual previous outputs (past states) in addition to the other inputs:

$$y(t) = f(w_1, w_2, \dots, w_p, b_1, b_2, \dots, b_p, y_{t-1}, y_{t-2}, \dots), \quad (5.17)$$

where $y(t)$ is the desired output, f is the neural network predictor (output of the MLP), w_i and b_i are network parameters, and y_{t-j} is the j^{th} previous output. In contrast, the parallel structure is used for predicting the current and future outputs of testing data, using current inputs and previous output states predicted by the neural network:

$$\tilde{y}_t = f(w_1, w_2, \dots, w_p, b_1, b_2, \dots, b_p, \tilde{y}_{t-1}, \tilde{y}_{t-2}, \dots), \quad (5.18)$$

where $\tilde{y}_{t-j}$ is the j^{th} previous predicted output. As illustrated in Figure 5.11, NARX uses previous actual outputs in the series-parallel structure and previous predicted outputs in the parallel structure. The choice of how many previous consecutive time steps to use as inputs depends on specific design and application requirements. Generally, increasing the number of previous time steps as inputs can improve network performance, but may also increase training time. This neural network structure leverages two key advantages: first, the output feedback loop in the learning and prediction process captures the mutual

information between sequential time steps, enhancing the ability to model temporal dependencies. Second, NARX networks have a simpler structure compared to other types of recurrent neural networks, resulting in reasonable prediction times.

Table 5.1: Hyperparameter configurations for NARX, LSTM, and Transformer models

Model	Hyperparameter Configuration
NARX	4 FC layers, 128 units each; Sigmoid activation; Input: 8 + 2 feedback; Output: 2 variables (V_x , V_y); Series-parallel (training), parallel (prediction)
LSTM	2 LSTM layers, 64 units each; Linear (output); Input: 8 + 2 feedback; Output: 2 variables (V_x , V_y); Window size = 20 time step
Transformer	2 attention layers per encoder/decoder; Embedding dim: 50; 2 Conv1D layers per encoder/decoder; 2 FF layers, 32/4 units each; Input: 8 + 2 feedback; Output: 2 variables (V_x , V_y)

Since this subsection compares different neural network models based on their estimation performance, both the training and testing datasets are derived from a single experimental domain: dry road conditions. All three neural network models were trained on the same dataset collected under dry road scenarios. For testing, two distinct maneuvers were used: (1) an acceleration-in-turn maneuver and (2) a high steering maneuver. Both test maneuvers were selected to ensure sufficient longitudinal and lateral velocity variations, providing a comprehensive basis for comparing the performance of the different methods. Table 5.1 presents the hyperparameters and the model structure for each neural network model used for comparison. Figure 5.12 presents the estimation results of the different methods for both longitudinal and lateral velocities during the acceleration-in-turn maneuver. As shown, both the LSTM and Transformer models achieve satisfactory performance in estimating these states, demonstrating their effectiveness at capturing temporal dependencies in time series prediction tasks. In contrast, the NARX model fails to accurately estimate future states, particularly for lateral velocity, where its output diverges after a certain period. A similar pattern is observed in Figure 5.13, which compares the three methods during the high steering maneuver. This behavior is expected, as both the LSTM and Transformer models are specifically designed for time series prediction.

Table 5.2 summarizes the estimation results for both test sets across the three methods, for both longitudinal and lateral velocities. The table reports training and testing times as well as estimation errors on the test sets. As shown, both the LSTM and Transformer models achieve low estimation errors on the test set, whereas the NARX model exhibits

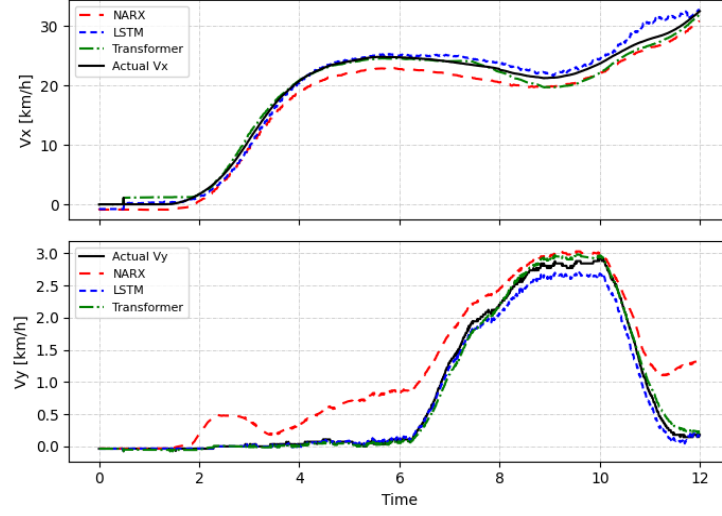


Figure 5.12: Comparison of the performance on V_x and V_y estimation of different NN-based models, on test 1.

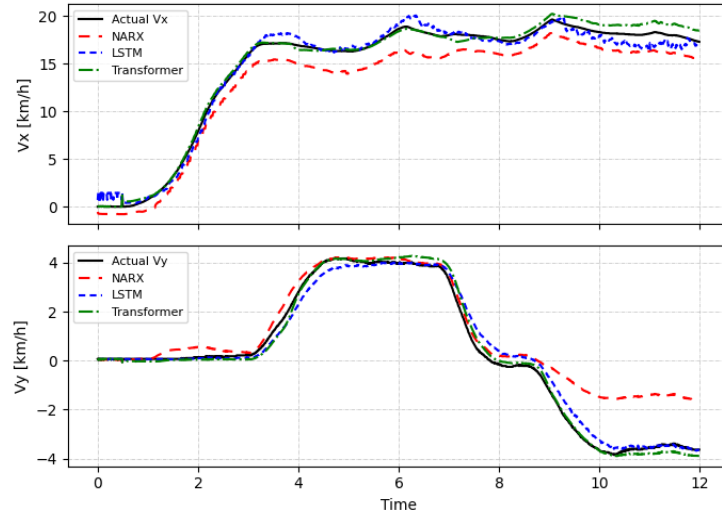


Figure 5.13: Comparison of the performance on V_x and V_y estimation of different NN-based models, on test 2

significantly higher errors. In terms of computational efficiency, the NARX model has the shortest training and testing times, due to its simple multilayer perceptron architecture. When comparing the two more advanced methods, the Transformer model demonstrates much lower training and testing times compared to the LSTM model. This is expected, as the LSTM’s complex cell structure leads to higher computational costs. Overall, the Transformer model offers the advantages of both low computational cost and high accuracy, making it a particularly suitable choice for these scenarios.

Table 5.2: Training times and estimation error (MAE) of the three models on two test sets.

Method	Training Time (s)	Error (Test 1)		Error (Test 2)	
		V_x	V_y	V_x	V_y
NARX	139.2	2.32	0.72	3.57	1.15
LSTM	370.0	0.46	0.18	0.71	0.36
Transformer	243.3	0.54	0.14	0.62	0.23

5.6 Summary

In this chapter, the proposed approach was extended for representative selection and continual learning to real-time state estimation. Specifically, a method was developed to effectively detect distribution shifts and to switch to a model-based estimator whenever the input data distribution differs from what the neural network has previously encountered. Additionally, after identifying the shift, the neural network is continuously updated using the new data to improve future estimation performance. Furthermore, a comparative study was conducted to evaluate the impact of using different neural network models for the state estimation task.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis contributes to the advancement in continual learning for time series forecasting by addressing the problem of catastrophic forgetting in neural network models, particularly within the context of vehicle state estimation. Two methods termed as ReSeleCT and EM-ReSeleCT, were developed to adapt the model to new information and preserve previous information, without the need to revisit the entire historical dataset. The proposed approaches achieve scalable and efficient learning by identifying and retaining only the most informative data points through representative selection.

In the univariate case, the proposed framework effectively uses statistical change point detection and memory-based optimization techniques to continually update of the neural network. This results in a model that not only maintains high predictive accuracy but also substantially reduces computational costs compared to traditional retraining methods. The method’s superiority over existing continual learning algorithms was validated across diverse experiments, demonstrating performance that closely matches the baseline with full retraining strategy, while requiring access to only a small subset of the old data. The framework was further extended to handle multivariate output scenarios. This setting introduces additional complexity in order to predict multiple states with one model, and presents challenges including redundancy in subset selection. By implementing the proposed method in an encoder-decoder Transformer architecture, this multi-output extension showed robust performance across a wide range of driving maneuvers and operating conditions. The results show that, the proposed method is capable of preventing catastrophic forgetting of the past information, while having significant reduction in training

time, compared to the Batch learning mode. With effective feature selection and sensitivity analyses strategies, the best set of input signals were selected, removing irrelevant signal measurements.

This thesis has also addressed the challenge of unknown task boundaries mostly occurs in non-stationary environments, by introducing a mechanism for automatic distribution shift detection. The system actively identifies operational changes and switches between model-based and data-driven estimators when required. This capability is essential for real-time deployment, allowing robust adjustment in non-stationary environments without manual intervention. The contributions of this work establish a unified framework for efficient, scalable, and adaptive continual learning in time series forecasting. By eliminating the reliance on full historical datasets, minimizing computational overhead, and allowing dynamic reconfiguration in response to environmental changes, the proposed method provides a practical and generalizable solution to real-world state estimation challenges in intelligent systems.

6.2 Future Work

There are several potential directions for future work that could further enhance the proposed continual learning method, and to better demonstrate its universality, scalability, and applicability of it, as described below:

Continual Learning Applications to Comprehensive Vehicle States: Future work could involve extending the proposed continual learning framework to a broader range of vehicle states, such as acceleration, braking, steering angle, and tire forces, in addition to velocity and position. In the context of vehicle state estimation, many critical states of the vehicle cannot be directly measured using available sensors. This requires accurate estimation techniques. Applying the method to a comprehensive set of vehicle state variables would allow for a more robust evaluation of its generalizability and effectiveness across different aspects of vehicle dynamics. This could also facilitate the development of integrated state estimation systems capable of real-time adaptation in complex and rapidly changing driving scenarios. Furthermore, such advancements could be instrumental for autonomous and semi-autonomous vehicle platforms, where reliable full-state awareness is essential for safe and efficient operation.

Through a comprehensive analysis of the multi-output representative selection method and its effectiveness in reducing redundancy among selected points, this study demonstrates a significant improvement in the training efficiency of neural network models—particularly

within vehicle datasets and related applications. Moreover, the methods developed in this work are closely related to state estimation problems in electric vehicles, such as the estimation of State of Charge (SoC) and State of Health (SoH) of batteries. The proposed approach can be extended to these domains as well, which are of growing importance in modern electric vehicle systems.

A practical deployment of the proposed continual learning method in vehicles as an estimation module, for critical vehicle safety tasks: Another promising direction for future work involves the practical deployment of the proposed continual learning framework in real-world vehicle systems. This includes implementing the state estimation algorithm as a modular component within an embedded system to enable real-time inference and adaptation on-board the vehicle. Such integration would require addressing additional challenges related to embedded systems and modules, including computational efficiency, memory constraints, and latency, while ensuring robustness under varying operating conditions. Ultimately, this step is crucial for advancing the proposed method toward its final goal of robust state estimation and practical deployment in autonomous and intelligent vehicle platforms.

Adaptation to Broader Domains including Biomedical, Environmental, and Forecasting Applications: While this thesis focuses on vehicle state estimation in experimental analysis and validation, it has potentials to apply in broader range of time series regression domains. In particular, in biomedical applications, many biomedical signals, such as electroencephalograms (EEG) and electrocardiograms (ECG), are high-dimensional, non-stationary, and often collected in streaming formats. Applying the ReSeleCT framework to such biomedical time series can enable personalized and adaptive health monitoring systems, capable of identifying anomalies or evolving patient conditions without retraining on large historical datasets. This is particularly valuable in intensive care settings, remote health monitoring, and wearable technologies. Furthermore, the continual learning approach can be extended to environmental and weather monitoring domains, where models must frequently incorporate new data to maintain accurate predictions under distribution shifts. These applications can benefit from continual adaptation to new data distributions while preserving the learned knowledge from past conditions. These slower-changing systems pose different challenges compared to automotive systems, such as larger memory windows and lower data frequency, which make them ideal benchmarks to assess the scalability and adaptability of the proposed method across varying temporal dynamics.

Extending the proposed method in to time series classification problems: A promising direction for future work is the adaptation of ReSeleCT and EM-ReSeleCT to time series classification problems. This can be achieved by leveraging the output of the

final hidden layer of the neural network before the classification layer (head), commonly used for text or image data. By identifying specific locations that provide critical information, representative data points that preserve essential information in streaming scenarios, necessary for classification tasks can be selected. Further research is needed to explore this approach, which could significantly broaden the applicability of the proposed method and enhance its utility in handling various types of data streams in both regression and classification scenarios.

References

- [1] MS Hossain Lipu, MA Hannan, Aini Hussain, Shaheer Ansari, SA Rahman, Mohamad HM Saad, and KM Muttaqi. Real-time state of charge estimation of lithium-ion batteries using optimized random forest regression algorithm. *IEEE Transactions on Intelligent Vehicles*, 8(1):639–648, 2022.
- [2] Yang Xing and Chen Lv. Dynamic state estimation for the advanced brake system of electric vehicles by using deep recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 67(11):9536–9547, 2019.
- [3] Hao Chen, Shanhe Lou, and Chen Lv. Hybrid physics-data-driven online modelling: Framework, methodology and application to electric vehicles. *Mechanical Systems and Signal Processing*, 185:109791, 2023.
- [4] Markus Schneider and Wolfgang Ertel. Robot learning by demonstration with local gaussian process regression. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 255–260. IEEE, 2010.
- [5] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [6] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
- [7] Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.
- [8] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric reviews*, 29(5-6):594–621, 2010.

- [9] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.
- [10] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.
- [11] Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: a survey. *IEEE computational intelligence magazine*, 4(2):24–38, 2009.
- [12] Bogdan Oancea and Ștefan Cristian Ciucu. Time series forecasting using neural networks. *arXiv preprint arXiv:1401.1333*, 2014.
- [13] Brian K Nelson. Time series analysis using autoregressive integrated moving average (arima) models. *Academic emergency medicine*, 5(7):739–744, 1998.
- [14] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International conference on machine learning*, pages 3925–3934. PMLR, 2019.
- [15] Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17:49–60, 2002.
- [16] Sethu Vijayakumar, Aaron D’souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.
- [17] Joyce Chen Yen Ngu, Wan Sieng Yeo, Teck Fu Thien, and Jobrun Nandong. A comprehensive overview of the applications of kernel functions and data-driven models in regression and classification tasks in the context of software sensors. *Applied Soft Computing*, page 111975, 2024.
- [18] Luka Peternel, Erhan Oztop, and Jan Babič. A shared control method for on-line human-in-the-loop robot learning based on locally weighted regression. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3900–3906. IEEE, 2016.
- [19] Muhammad Tufail, Shahzad Anwar, Zubair Ahmad Khan, and Muhammad Tahir Khan. Real-time impedance control based on learned inverse dynamics. *Arabian Journal for Science and Engineering*, 45:5043–5055, 2020.

- [20] Grady R Williams, Brian Goldfain, Keuntaek Lee, Jason Gibson, James M Rehg, and Evangelos A Theodorou. Locally weighted regression pseudo-rehearsal for adaptive model predictive control. In *Conference on Robot Learning*, pages 969–978. PMLR, 2020.
- [21] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.
- [22] Cheng Qian, Ning He, Lile He, Ruoxia Li, and Fuan Cheng. A weighted gaussian process regression model based on improved local outlier factor and its application in state of health estimation of lithium-ion battery. *Engineering Applications of Artificial Intelligence*, 138:109314, 2024.
- [23] Armin Lederer, Alejandro J Ordóñez Conejo, Korbinian A Maier, Wenxin Xiao, Jonas Umlauf, and Sandra Hirche. Gaussian process-based real-time learning for safety critical applications. In *International Conference on Machine Learning*, pages 6055–6064. PMLR, 2021.
- [24] John Platt. A resource-allocating network for function interpolation. *Neural computation*, 3(2):213–225, 1991.
- [25] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.
- [26] Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE transactions on neural networks*, 16(1):57–67, 2005.
- [27] Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006.
- [28] Ying Mei, Chengbo Lu, Chenghao Tang, Yonghua Wu, and Guoqing Meng. A storage and classification algorithm for concept drift data streams based on os-elm. *Journal of Classification*, pages 1–21, 2025.
- [29] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

- [30] Wei Cong, Yang Cong, Gan Sun, Yuyang Liu, and Jiahua Dong. Self-paced weight consolidation for continual learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(4):2209–2222, 2023.
- [31] Yung-I Chang and Rih-Teng Wu. Recognition of structural components and surface damage using regularization-based continual learning. *Structural Control and Health Monitoring*, 2025(1):6005674, 2025.
- [32] Yujiang He and Bernhard Sick. Clear: An adaptive continual learning framework for regression tasks. *AI Perspectives*, 3(1):1–16, 2021.
- [33] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- [34] Ju Xu, Jin Ma, Xuesong Gao, and Zhanxing Zhu. Adaptive progressive continual learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):6715–6728, 2021.
- [35] Jiaqi Yao, Bowen Zheng, and Julia Kowal. Continual learning for online state of charge estimation across diverse lithium-ion batteries. *Journal of Energy Storage*, 117:116086, 2025.
- [36] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001.
- [37] Yi Zhang, Hu Chen, Wenjun Xia, Yang Chen, Baodong Liu, Yan Liu, Huaiqiang Sun, and Jiliu Zhou. Learn++: recurrent dual-domain reconstruction network for compressed sensing ct. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 7(2):132–142, 2022.
- [38] Eugene Li, Yang Li, Sanjeev Bedi, William Melek, and Paul Gray. Incremental learning of lstm-autoencoder anomaly detection in three-axis cnc machines. *The International Journal of Advanced Manufacturing Technology*, 130(3):1265–1277, 2024.
- [39] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.
- [40] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

- [41] Vibhor Gupta, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Continual learning for multivariate time series tasks with variable input dimensions. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 161–170. IEEE, 2021.
- [42] Reem A Mahmoud and Hazem Hajj. Multi-objective learning to overcome catastrophic forgetting in time-series applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(6):1–20, 2022.
- [43] Suguru Kanoga, Ryo Karakida, Takayuki Hoshino, Yuto Okawa, and Mitsunori Tada. Deep generative replay-based class-incremental continual learning in semg-based pattern recognition. In *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1–4. IEEE, 2024.
- [44] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*, 2021.
- [45] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc’Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.
- [46] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in neural information processing systems*, 33:14879–14890, 2020.
- [47] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8218–8227, 2021.
- [48] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.
- [49] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33:1023–1035, 2020.
- [50] Jacob Armstrong and David A Clifton. Continual learning of longitudinal health records. In *2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 01–06. IEEE, 2022.

- [51] Andrea Cossu, Antonio Carta, Vincenzo Lomonaco, and Davide Bacciu. Continual learning for recurrent neural networks: an empirical evaluation. *Neural Networks*, 143:607–627, 2021.
- [52] Quentin Besnard and Nicolas Ragot. Continual learning for time series forecasting: A first survey. *Engineering Proceedings*, 68(1):49, 2024.
- [53] Yujiang He. Adaptive explainable continual learning framework for regression problems with focus on power forecasts. *arXiv preprint arXiv:2108.10781*, 2021.
- [54] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.
- [55] Xu Chen, Junshan Wang, and Kunqing Xie. Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning. *arXiv preprint arXiv:2106.06273*, 2021.
- [56] Daniel Philps, Tillman Weyde, Artur d’Avila Garcez, and Roy Batchelor. Continual learning augmented investment decisions. *arXiv preprint arXiv:1812.02340*, 2018.
- [57] Gozde O Sahinoglu, Milutin Pajovic, Zafer Sahinoglu, Yebin Wang, Philip V Orlik, and Toshihiro Wada. Battery state-of-charge estimation based on regular/recurrent gaussian process regression. *IEEE Transactions on Industrial Electronics*, 65(5):4311–4321, 2017.
- [58] Qin Li, Hongwen He, Xiaokai Chen, and Jianping Gao. Learning-based vehicle state estimation using gaussian process regression combined with extended kalman filter. *Journal of the Franklin Institute*, 361(9):106907, 2024.
- [59] Bruno Henrique Groenner Barbosa, Nan Xu, Hassan Askari, and Amir Khajepour. Lateral force prediction using gaussian process regression for intelligent tire systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(8):5332–5343, 2021.
- [60] Zhuoping Yu, Xinchun Hou, Bo Leng, and Yuyao Huang. Mass estimation method for intelligent vehicles based on fusion of machine learning and vehicle dynamic model. *Autonomous Intelligent Systems*, 2(1):4, 2022.
- [61] Metin Yılmaz, Eyüp Çinar, and Ahmet Yazıcı. A transformer-based model for state of charge estimation of electrical vehicle batteries. *IEEE Access*, 2025.

- [62] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [63] Chenkai Tan, Yingfeng Cai, Hai Wang, Xiaoqiang Sun, and Long Chen. Vehicle state estimation combining physics-informed neural network and unscented kalman filtering on manifolds. *Sensors*, 23(15):6665, 2023.
- [64] Rongye Shi, Zhaobin Mo, and Xuan Di. Physics-informed deep learning for traffic state estimation: A hybrid paradigm informed by second-order traffic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 540–547, 2021.
- [65] Arvin Hosseinzadeh, Reza Valiollahi Mehrizi, Mohammad Pirani, Shojaeddin Chenouri, and Amir Khajepour. Reselect: A new approach for continual learning with application to vehicle state estimation. *IEEE Transactions on Intelligent Vehicles*, 10(3):2103–2114, 2025.
- [66] Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. Retrieve: Core-set selection for efficient and robust semi-supervised learning. *Advances in Neural Information Processing Systems*, 34:14488–14501, 2021.
- [67] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. Gcr: Gradient coreset based replay buffer selection for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 99–108, 2022.
- [68] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.
- [69] Rafal Baranowski, Yining Chen, and Piotr Fryzlewicz. Narrowest-over-threshold detection of multiple change points and change-point-like features. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 81(3):649–672, 2019.
- [70] Lars Imsland, Tor A Johansen, Thor I Fossen, Håvard Fjær Grip, Jens C Kalkkuhl, and Avshalom Suissa. Vehicle velocity estimation using nonlinear observers. *Automatica*, 42(12):2091–2103, 2006.
- [71] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [72] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.
- [73] Arvin Hosseinzadeh, Ladan Khoshnevisan, Mohammad Pirani, Shojaeddin Chenouri, and Amir Khajepour. An efficient continual learning framework for multivariate time series prediction tasks with application to vehicle state estimation. *arXiv preprint arXiv:2503.01669*, 2025.
- [74] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.
- [75] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.
- [76] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.
- [77] Jaime Pizarroso, José Portela, and Antonio Muñoz. Neuralsens: sensitivity analysis of neural networks. *Journal of Statistical Software*, 102:1–36, 2022.
- [78] Arvin Hosseinzadeh, Mohammadreza Ghorbani, Ladan Khoshnevisan, Mohammad Pirani, Shojaeddin Chenouri, and Amir Khajepour. Adaptive state estimation and continual learning under data distribution shift. *TechRxiv, preprint*, 2025.