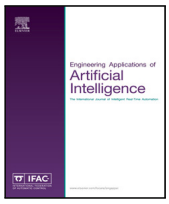
Engineering Applications of Artificial Intelligence 150 (2025) 110459

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

Continual Learning for Behavior-based Driver Identification

Mattia Fanan, Davide Dalle Pezze *, Emad Efatinasab, Ruggero Carli, Mirco Rampazzo, Gian Antonio Susto

University of Padova, Padova, Italy

ARTICLE INFO

Keywords:
Continual Learning
Deep Learning
Driver Identification

ABSTRACT

Behavior-based Driver Identification is an emerging technology that recognizes drivers based on their unique driving behaviors, offering important applications such as vehicle theft prevention and personalized driving experiences. However, most studies fail to account for the real-world challenges of deploying Deep Learning models within vehicles. These challenges include operating under limited computational resources, adapting to new drivers, and changes in driving behavior over time. The objective of this study is to evaluate if Continual Learning (CL) is well-suited to address these challenges, as it enables models to retain previously learned knowledge while continually adapting with minimal computational overhead and resource requirements. We tested several CL techniques across three scenarios of increasing complexity based on a well-known dataset for the Driver Identification problem. This work provides an important step forward in scalable driver identification solutions, demonstrating that CL approaches, such as Dark Experience Replay (DER), can obtain strong performance with only an 11% reduction in accuracy compared to the static scenario. Furthermore, to enhance the performance, we propose two new methods, Smooth Experience Replay (SmooER) and Smooth Dark Experience Replay (SmooDER), that leverage the temporal continuity of driver identity over time to enhance classification accuracy. Our novel method, SmooDER, achieves optimal results with only a 2% accuracy reduction compared to the 11% of the DER approach. In conclusion, this study proves the feasibility of CL approaches to address the challenges of Driver Identification in dynamic environments, making them suitable for deployment on cloud infrastructure or directly within vehicles.

1. Introduction

Behavior-based driver identification uses driving patterns—such as acceleration, steering angles, and braking patterns over time—extracted from vehicle sensor data to determine the driver's identity. After traditional authentication methods, including physical and wireless keys, have been proven to be vulnerable (Garcia et al., 2016; Francillon et al., 2011), researchers have proposed behavior-based approaches in order to tackle the problem (Kang et al., 2019; Ravi et al., 2022; Tseng et al., 2023). The more recent trend has seen the increasing use of Deep Learning techniques to improve accuracy as well as robustness (Lin et al., 2018; Girma et al., 2019; Xun et al., 2019; El Mekki et al., 2019; Gahr et al., 2019; Azadani and Boukerche, 2020; Abu-Gellban et al., 2020). These solutions provide significant advantages in solving the ongoing issue of *car theft*, which is still a problem despite many technological advances. Another advantage is that car rental companies and fleet managers can ensure only authorized people drive their vehicles, improving efficiency. It can also enable customized *in-car experiences*, better *roadside assistance*, and optimized *maintenance schedules*.

While the literature extensively explores the potential applications of Behavior-based Driver Identification systems, challenges specifically related to real-world implementation are often overlooked. Many of these systems depend on Machine Learning (ML) and Deep Learning (DL) models for classification, which are generally trained offline using large-scale datasets of drivers. Although efficient, collecting enough training data requires a considerable amount of time, leaving vehicles vulnerable to theft during the process. Efatinasab et al. (2024, 2022). Additionally, in scenarios like car sharing or rental services, a complete retraining policy is impractical. Since the model would have to learn all the data every time a new driver is added, straining resources like the GPU and memory. Even with cloud storage, the time required and the costs can be prohibitive. Lastly, Fine-Tuning the model using only new data fails to address the issue, as it can lead to Catastrophic Forgetting (CF), where the model's performance on previously learned tasks deteriorates as it adapts to new information (De Lange et al., 2021; McCloskey and Cohen, 1989).

To address these challenges, we propose applying Continual Learning (CL) techniques to the Driver Identification problem. Being the

* Corresponding author.
E-mail address: davide.dallepezze@unipd.it (D. Dalle Pezze).

https://doi.org/10.1016/j.engappai.2025.110459
Received 30 November 2024; Received in revised form 26 January 2025; Accepted 26 February 2025
Available online 20 March 2025

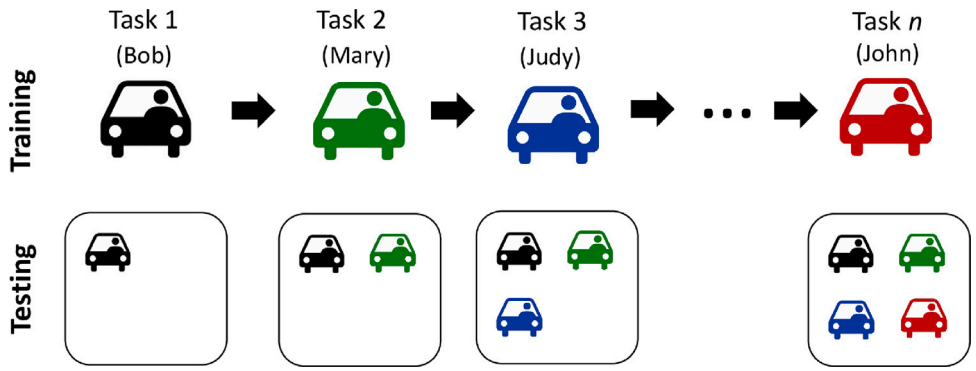


Fig. 1. General representation of the experimental setup. Since the goal is to learn new tasks as they arrive while retaining knowledge from previous ones, the test set for each task consists of all the test data from drivers whose labels already appeared.

primary goal of CL to adapt to incoming data while preserving previously acquired knowledge, it is specifically designed to tackle the issue of Catastrophic Forgetting using resources comparable to Fine-Tuning. Therefore, we study scenarios where the models learn incrementally from new drivers (see Fig. 1 for the general framework). Specifically, we propose an evaluation across three scenarios of increasing complexity and realism, based on the well-known OCSLab dataset (Martinelli et al., 2020). The first scenario represents a classical CL setting, while the other two modify how the data is presented to the system, making them increasingly reflective of real-world conditions. Interestingly, making the simulations more realistic brings both challenges and opportunities. The driver, for example, is not expected to change every few minutes. This continuity is leveraged in this work to introduce two new methods, SmooER and SmooDER, which aim to further narrow the performance gap compared to complete retraining.

This work provides an important step forward in scalable driver identification solutions, demonstrating that CL approaches, such as DER, can obtain strong performance, with only an 11% reduction in accuracy compared to the static scenario. Furthermore, our novel method, SmooDER, achieves optimal results with only a 2% reduction compared to the 11% of the DER approach. Such results prove the feasibility of CL approaches to address the challenges of Driver Identification in dynamic environments, making them suitable for deployment on cloud infrastructure or directly within vehicles. Additionally, we have made our implementation publicly available,[1] for enhanced reproducibility and to facilitate future research in the field.

Our contributions are summarized as follows:

- We investigate the Driver Identification problem in the CL framework.
- We propose and implement on the OCSLab dataset three new scenarios that progressively increase in complexity to incorporate additional challenges and make them more similar to a real application of Driver Identification in a physical vehicle.
- We evaluate the efficacy of CL techniques in the three scenarios, proving the feasibility of their deployment on cloud infrastructure or directly within vehicles.
- We propose two novel methods, SmooER and SmooDER, that exploit the driver ID continuity expected in deployment, achieving superior results compared to the other CL approaches.

The rest of this article is organized as follows: Section 2 reviews the literature on the Driver Identification problem and the CL framework. Section 3 describes our proposed scenarios, the CL approaches we tested, and our new methods, SmooDER and SmooER. Section 4 details the experimental setup, including the dataset, training configuration, and model hyperparameters. Section 5 presents the results for each scenario. Finally, Section 6 discusses the findings and suggests future research directions in this field.

[1] https://github.com/MattiaFanan/CL-DriverIdentification.

2. Related work

In Section 2.1, we review the literature on behavior-based driver identification. Following that, Section 2.2 will provide an overview of the Continual Learning topic, highlighting the main scenarios and methods in the field.

2.1. Driver identification

A significant amount of research addresses the challenge of identifying the driver based on behavioral data. This has several practical applications, including preventing vehicle theft and improving personalized driving experiences. Collecting and analyzing drivers' behavioral data enables the optimization of driver-based insurance schemes by dynamically calculating risk factors. Cura et al. (2021) and Hill (2024). Moreover, it would enable the personalization of the driving experience by adjusting vehicle settings based on user preferences, as demonstrated in previous studies. El Mekki et al. (2019).

Most methodologies consider the Controller Area Network (CAN) bus data via the On-Board Diagnostics (OBD-II) port, utilizing its variety of sensors and actuators to assess the vehicle's condition. Specifically, most studies rely on the publicly available OCSLab dataset (Martinelli et al., 2020; Kwak et al., 2016). It features 54 sensors from the CAN bus and includes data from ten drivers. Other datasets, although less commonly used, incorporate different data types, such as Global Positioning System (GPS) (Rahim et al., 2019) data and stability measurements (Ahmadi-Assalemi et al., 2021).

Many studies use Artificial Intelligence (AI) and ML algorithms to tackle Driver Identification. The framework proposed in Xun et al. (2020) addresses the problem by analyzing the driving dynamics signals that can be acquired from commercially available vehicles. The data gathered from the sensors is filtered and pre-processed into sliding windows to allow the extraction of meaningful statistical and spectral features. Then, a driver is associated with every window, and these individual predictions are finally aggregated to obtain a unified result for the entire trip. Erzin et al. (2006) explore the amalgamation of various features, including pedal pressure, vehicle speed, engine speed, and steering angle, to discern the driver identity.

Various works have employed classical machine learning algorithms, such as Random Forest (RF) (Ahmadi-Assalemi et al., 2021; Hallac et al., 2016; Ezzini et al., 2018; Rahim et al., 2019), Support Vector Machine (SVM) (Hallac et al., 2016; Ezzini et al., 2018; Rahim et al., 2019; Marchegiani and Posner, 2018; Burton et al., 2016; Azadani and Boukerche, 2021), K-Nearest Neighbors (KNN) (Kwak et al., 2016; Lin et al., 2018; Hallac et al., 2016; Ezzini et al., 2018; Rahim et al., 2019), and K-means (Kang et al., 2019). Marchegiani and Posner (2018) leverage CAN bus data from an electric vehicle to study pedal operation patterns and GPS traces across different drivers on the same route. Their research aims to develop a user authentication

system using techniques, such as SVM and Universal Background Model (UBM).

Furthermore, there is increasing emphasis in the literature on the use of Recurrent Neural Networks (RNNs) (El Mekki et al., 2019; Gahr et al., 2019; Zhang et al., 2019; Girma et al., 2019), including Long Short-Term Memory networks (LSTMs) (Azadani and Boukerche, 2020; Ravi et al., 2022; Abu-Gellban et al., 2020; El Mekki et al., 2019). Their widespread adoption underscores their effectiveness in capturing temporal dependencies and modeling sequential data, thereby improving the accuracy and reliability of behavior-based driver identification systems. Xun et al. (2019) propose driver fingerprinting, utilizing CAN bus data for real-time user authentication. Their method employs a combination of Convolutional Neural Networks (CNN) and Support Vector Domain Description (SVDD) to detect unauthorized drivers. The authors in Chen et al. (2019) introduce an unsupervised method using a three-layer non-negativity-constrained autoencoder to find the best sliding window size. They then employ a Deep Autoencoder to extract hidden driving behavior features for better driver identification.

2.2. Continual learning

Commonly, a Machine Learning model is built on the premise that the training dataset is representative of the reality one intends to model. However, hidden factors can change the observable data-generating distribution of the environment, causing a performance decay during the deployment. Collecting another dataset and re-training the model is not a definitive solution if the previously acquired knowledge is not preserved. In fact, additional training on new data (Fine-Tuning) causes the effect known as Catastrophic Forgetting, where the model's performance on previous data deteriorates (De Lange et al., 2021; McCloskey and Cohen, 1989). Continual Learning overcomes this limitation by enabling models to learn from new data without forgetting previously learned information. This framework has been successfully applied across a diverse range of real-world applications. These span from quality inspection (Xie et al., 2024; Bugarin et al., 2024) and medical diagnostics (Ceccon et al., 2024; Singh et al., 2023) to autonomous driving (Verwimp et al., 2023; Shieh et al., 2020) and robotics (Qiao et al., 2024; Pasti et al., 2024).

2.2.1. Continual learning scenarios

CL typically aims to learn a sequence of tasks, with each task representing a classification problem. Many works organized similar experimental setups into the so-called CL scenarios. Essentially they are the definition of what aspect of the problem is allowed to change between the tasks. The vast majority of the published works use one of the following: *Task-Incremental Learning* (TIL) (van de Ven and Tolias, 2019), involves training a model on a sequence of general tasks but relies on the availability at the deployment time of an identifier to let the model know which of the learned tasks it must use as a reference to make the inference. When such information is not available one can rely on either *Class-Incremental Learning* (CIL) (van de Ven and Tolias, 2019) or *Domain-Incremental Learning* (DIL) (van de Ven and Tolias, 2019). The first only allows for subsequent tasks with a disjoint set of classes. In contrast, the second requires tasks with the same set of classes but allows a domain shift. Lastly, another important scenario to model more complex environments is *New Instances & Classes* (NIC) (Lomonaco and Maltoni, 2017), which allows updating the model knowledge with both the introduction of additional classes and domain shifts (without the availability of the task identifier).

2.2.2. Continual learning approaches

In the CL literature, the methods can be grouped into three big families known as rehearsal-based, regularization-based, and architecture-based.

Rehearsal-based techniques relay on storing and reusing past data samples during training. While various approaches exist in this category (Tiwari et al., 2022; Lopez-Paz and Ranzato, 2017), one of the most renowned methods is Experience Replay (ER) (Rolnick et al., 2019), also known as Replay. With this method, samples from the Replay memory (which stores data from previous tasks) are combined with data from the new task (Merlin et al., 2022). This allows the model to retain accumulated knowledge by periodically revisiting past task instances while learning new information. It has demonstrated remarkable performance in multi-class classification problems under all three CL scenarios (DIL, CIL, TIL).

Regularization-based approaches incorporate additional constraints or penalties during training to preserve the knowledge of previous tasks. Two well-known methods belonging to this category are Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) and Learning without Forgetting (LwF) (Li and Hoiem, 2017). The idea behind EWC is to assign higher penalties to changes in parameters that are important for previously learned tasks, thus protecting them from significant updates during the training of the next tasks.

LwF is based on the concept of *Knowledge Distillation*, introduced by Hinton et al. (2015). The loss function incorporates both the classification error for the current task and a distillation loss, which uses the previous model's output as soft labels. This approach encourages the model to produce consistent outputs across both old and new tasks, helping it retain prior knowledge while learning new information.

Architecture-based approaches aim to modify the original model's architecture by adding new parameters, which help redirect the optimizer updates away from the old ones that contain the accumulated knowledge. The specific approach used to achieve this varies widely (Rusu et al., 2016; Fernando et al., 2017; Mallya and Lazebnik, 2018). However, these methods generally suffer from two major drawbacks. Firstly, most require the task ID at inference time, which is unavailable in our scenarios. More importantly, they operate by design on the structure of the underlying model, making it larger after every task, which also leads to increases in latency at inference time.

Furthermore, the CL literature explores methods that combine multiple strategies to improve performance and robustness (Soutif-Cormerais et al., 2023). The DER method (Buzzega et al., 2020) is one such example that integrates the Replay memory strategy with the use of logits from previous models, as done by LwF.

3. Methodology

In Section 3.1, we define the methodological framework designed to integrate Continual Learning into behavior-based driver identification systems. Then in Section 3.2, three progressively realistic scenarios are introduced to evaluate the CL methods in the context of driver identification. In Section 3.3 we present the CL methods used in our experiments. Finally, in Section 3.4 our proposed approaches, SmooDER and SmooER, are described.

3.1. Problem definition

As mentioned, Driver Identification is an important problem with numerous real-world applications. However, despite existing methods achieving impressive results, they struggle to efficiently acquire new knowledge over time, as retraining a new model from scratch using all the seen data is resource-intensive and impractical. This limitation raises concerns about the feasibility of implementing them in real-world scenarios due to their scalability issues.

In a generic scenario, a model is trained on a sequence of T tasks. Each task t is associated with a dataset $D_t = (X_t, Y_t)$. Each training instance $x_{it} \in \mathbb{R}^{W \times F}$ in $X_t = [x_{1t}, \ldots, x_{nt}]$ is a matrix recording the values of the F features of interest across W consecutive time steps. The corresponding labels are stored in $Y_t = [y_{1t}, \ldots, y_{nt}]$ with $y_{it} \in C_t$, where C_t is the set of classes for task t.

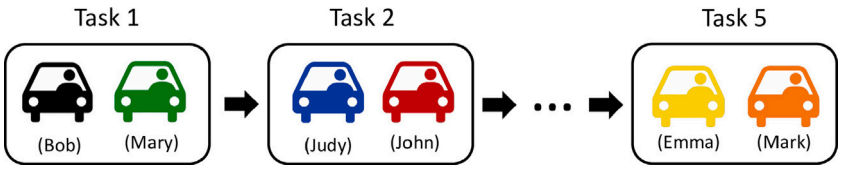


Fig. 2. Task sequence representation for Scenario 1. Given ten drivers, each task contains information about two new drivers, for a total of five tasks.

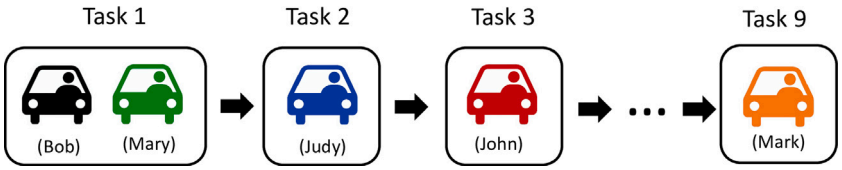


Fig. 3. Task sequence representation for Scenario 2. A single driver is added to the system each time (except for task 1).

The objective of the model at task t, f_{θ_t}, is to learn a mapping:

$$f_{\theta_t} : x \rightarrow y \quad \text{such that} \quad x \in \mathbb{R}^{W \times F}, \quad y \in \bigcup_{i=1}^{t} C_i \tag{1}$$

where $\bigcup_{i=1}^{t} C_i$ is the set of all classes encountered up to and including task t. Therefore, after training the model for task t, it is expected to continue performing well on previous tasks as well.

3.2. Scenarios

We propose and study the following three CL scenarios in the context of Driver Identification, each introducing a progressively higher level of complexity and realism. The goal in all cases is to train the underlying model to recognize new drivers accurately while preserving its ability to distinguish those learned previously.

3.2.1. Scenario 1 - two new drivers

The first scenario is the simplest one. The stream is created similarly to those in classic CL benchmarks like Split-MNIST and Split-CIFAR10, where the dataset is split into 5 tasks, each composed of data from only two classes (van de Ven and Tolias, 2019). This scenario replicates the typical conditions under which CL techniques are usually proposed. Therefore, we split the OCSLab dataset into five tasks, each composed of two drivers' data (see Fig. 2). Since for each task, two new classes are added, it fits the CIL setting (described in Section 2.2).

3.2.2. Scenario 2 - one new driver

In practice, there is no particular reason to add authorized drivers in pairs like in Scenario 1. However, in cases like a family car, it is reasonable to expect that the owner might need to authorize an additional driver, such as when a child obtains their driver's license. Therefore, for Scenario 2 we have a sequence of classes as follows: 2,1,1,1,1,1,1,1, for a total of 9 tasks (see Fig. 3). The first task is composed of two classes since evaluating accuracy on a single class at the start would be pointless. Compared to Scenario 1, here more tasks are required to cover the same number of classes (9 instead of 5). As a result, more updates are needed to learn the same number of drivers, which can potentially increase forgetting. Additionally, introducing one class at a time makes it more challenging for the model to learn disjoint representations for each driver, as they are learned separately. Since a new class is added with each task, this scenario fits the CIL setting, as described in Section 2.2.

3.2.3. Scenario 3 - two new sessions

Here, we outline a scenario that we believe is the most suitable for the real-world implementation of a Driver Identification solution.

A critical aspect of adding a new user to the system is the process of collecting the training data, since the Behavior-based Driver Identification system must be temporarily disabled, relying only on conventional detection. Traditional data collection methods, such as those used in the OCSLab dataset, typically involve planning lengthy routes for each driver, with multiple sessions needed to accurately capture individual styles. This process is very demanding, often requiring hours of driving, and lacks flexibility. The result is a sequence of driving sessions from various drivers, often with repeated sessions for consistency.

To alleviate this burden, we propose a system in which the conventional and behavior-based authentication components complement each other. When introducing a new user, a startup phase defined by a super-user relies exclusively on conventional identification. After this initial phase, the system can automatically store a buffer of data from subsequent driving sessions. If the predictions from the conventional and behavior-based identification converge, the collected data can be utilized for additional training to adapt to domain shifts. Conversely, discrepancies can trigger an error notification. The most efficient time to initiate training using the buffered data is when the car is parked for an extended period, avoiding unnecessary resource usage while driving. This approach can be naturally represented using the OCSLab dataset by segmenting into tasks a random permutation of the driving sessions. Only tasks involving the introduction of a new driver require access via traditional identification methods, enhancing security in the others.

We propose using two driving sessions per task, as depicted in Fig. 4. In real-world scenarios, this could represent either a short round trip by a single driver or a longer journey involving a driver change or a stop. Given that the dataset contains 20 driving sessions (two per driver), this setup corresponds to a total of 10 tasks.

Unlike the previous scenarios, which fall under the CIL setting, this scenario aligns with the NIC setting, as described in Section 2.2. Here, tasks may include new driving sessions from previously seen drivers or data from entirely new drivers, making it a dynamic and realistic representation of the driver identification problem.

3.3. Methods

Our goal is to validate, across three distinct scenarios, whether CL techniques can effectively address the Driver Identification problem. Therefore, we select three baselines, Fine-Tuning as the lower bound, and Joint Training and Cumulative as the upper bounds for model performance. Additionally, we test several well-known CL techniques such as ER, EWC, LwF, and DER.

- **Joint Training**: The model is trained on all classes simultaneously, representing the traditional static training approach. This serves as an upper baseline for CL strategies, as it is not affected by Catastrophic Forgetting.

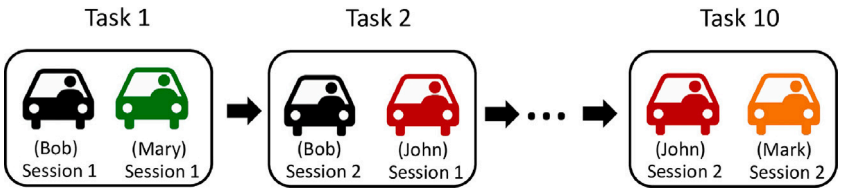


Fig. 4. Task sequence representation for Scenario 3. The data is divided into drive sessions (there are two of them for each driver) and each task contains the data of two sessions. As a result, the next task can either introduce new drivers or expand the knowledge of previously encountered ones.

- **Cumulative**: This represents an additional upper bound of the CL techniques. The model is trained sequentially on the stream of tasks. However, contrary to Fine-Tuning, the model at each task is trained with all data seen so far. The Cumulative approach represents the current solution for adding additional users; however, it presents scalability challenges when expanding to a larger number of users.
- **Fine-Tuning**: The model is presented sequentially with data from the current task. No particular technique maintains the model's knowledge, so it is considered a lower bound for the CL approaches.
- **ER**: Experience Replay creates a memory to store samples from previous tasks (Rolnick et al., 2019). We consider a fixed memory size, balancing the space reserved for each class and task. When new data needs to be added to a full memory, a random selection of points is equally removed from each class allocation to make space.
- **EWC**: Elastic Weight Consolidation relies on the Fisher information matrix to identify which neural network parameters are important for the tasks the model has already learned (Kirkpatrick et al., 2017). The Fisher information matrix measures the sensitivity of the loss function to changes in the model parameters. The importance is used to make the important weights more rigid.
- **LwF**: Learning without Forgetting is the most well-known technique among the distillation-based approaches (Li and Hoiem, 2017). In our implementation, at each training batch, a distillation loss is computed as the Mean Squared Error (MSE) between the logits of the teacher and the student. The distillation loss is then added to the model loss for the task.
- **DER**: Dark Experience Replay (Buzzega et al., 2020) is similar to ER as it keeps a small sample from the previous tasks to refresh the old knowledge when learning from new data, but instead of storing the ground truth labels, it stores the so-called "dark knowledge" generated by a previous version of the model (its output logits). The original paper proposes another variant of DER termed DER++, which stores both ground truth and the model output logits for each saved sample.

3.4. Prediction smoothing

With this work, we introduce two new CL methods derived from DER++ and ER, specifically Smooth Dark Experience Replay (SmooDER) and Smoot Experience Replay (SmooER). These methods are designed to leverage a key feature of the scenarios we consider: the natural continuity of the driver in real-world applications.

We needed a mechanism to correct occasional classification errors due to forgetting without causing significant delays during driver changes. Given that the labels are expected to change infrequently, smoothing the model's output enhances robustness, as multiple consecutive incorrect predictions are needed to significantly alter the integrated result. To achieve this goal we used a causal rolling-window average of the model's output. This approach, inspired by temporal smoothing techniques, ensures that predictions are more stable over short periods of time, thereby aligning better with real-world scenarios where high-frequency label transitions are rare.

Formally, let W be the window size, k the number of classes currently present, and f_θ be the current model. During inference, given the ith sample x_i, the model calculates its output $z_i = f_\theta(x_i) = [z_{i1}, z_{i2}, \dots, z_{ik}]$, where z_{ij} represents the confidence the model has about ith sample being a member of the jth class or not.

We then apply the causal rolling-window average of Eq. (2) (where W is the smoothing window size) to the outputs, resulting in $\widetilde{z}_i = [\widetilde{z}_{i1}, \widetilde{z}_{i2}, \dots, \widetilde{z}_{ik}]$, which ensures that occasional misclassifications do not cause abrupt changes in the output. Finally, the final output is determined by selecting the class with the highest confidence, as shown in Eq. (3).

$$\widetilde{z}_{ij} = \frac{1}{W} \sum_{r=i-W+1}^{i} z_{rj} \tag{2}$$

$$\hat{y}_i = \underset{j \in [1 \cdots k]}{\operatorname{argmax}} \widetilde{z}_{ij} \tag{3}$$

A larger window is advantageous for noisy predictors, as it can suppress more misclassifications. Conversely, a smaller window suits a robust classifier, reducing errors during driver changes, as fewer new-driver predictions are needed to alter the output.

In this way, our methods, SmooER and SmooDER, exploit the temporal correlation of the driver identifier by incorporating a smoothing technique, making the system more resilient to isolated classification errors while maintaining the independence of the underlying base CL method's parameters and mechanism.

4. Experimental setting

In the following we present the experimental setting to evaluate the performance of CL methods in the context of behavior-based driver identification. Specifically, Section 4.1 presents the OCSLab dataset along with details on the applied preprocessing. In Section 4.2, we describe the Machine Learning model that forms the foundation of our approach, as well as the details of the training procedure. Section 4.3 provides an overview of the hyperparameters for each CL method, explaining their purpose and presenting the optimal values identified in our experiments. Finally, in Section 4.4 we describe the metric used to compare the results of our work.

4.1. Specifics on the OCSLab dataset

The dataset used in our tests comprises data from an experiment involving ten drivers (labeled from "A" to "J"), each of them completed two round trips under varied road conditions driving the same car, for a total of about 23 hours of driving. The dataset contains the recording of 51 sensors from the OBD2 port with a sampling frequency of 1 Hz (you can refer to the original paper (Martinelli et al., 2020) for more detail and the complete list of sensors).

Our main objective is to test the applicability of CL techniques in the Driver Identification problem. Since CL is generally model-agnostic, we decided to build our system over one of the most cited works in this field (Zhang et al., 2019).

To preprocess our data, we removed the features with zero variance in the dataset and then used the mean and standard deviation normalization. This technique standardizes the features of a dataset,

transforming each feature x using the formula $x' = \frac{x-\mu}{\sigma}$, where μ is the mean and σ is the standard deviation. This process centers the data around zero and scales it to have a unit variance, improving the performance and convergence of many machine learning algorithms.

The next crucial part in preprocessing data for a sequence classification task is to choose suitable parameters for the windowing of our data.

- The **window length** is how many consecutive samples a training instance is made of and consequently defines the complexity of the task to be learned as well as the richness in the causal relationship that the model can exploit.
- The **stride of the window** refers to the displacement between consecutive windows in the data, allowing for potential overlap. This parameter is crucial as it determines how often each data point appears in the final dataset. A smaller stride means more frequent appearances of the same point in different contexts, enhancing model generalization. However, it also increases the dataset size, which can extend training time. Note that a larger stride does not reduce model responsiveness during deployment, as in the real implementation it is possible to generate a new window with only one new sample.

To set these parameters, we used the configuration from the reference paper with the shortest window length and stride to create a less complex but more responsive model. Specifically, we set the window length to 60 and the stride to 6.

4.2. Models and training

We chose LSTM as our model because it consistently performed the best in the reference paper's result tables (Zhang et al., 2019). According to their description, the network structure is a two-layer LSTM, with each layer having 128 hidden states and a dropout rate of 0.5.

LSTMs are sequence-to-sequence models that transform the input into a sequence of hidden states (Hochreiter, 1997). To create a classifier, a fully connected layer is applied to the last hidden state of the final layer with a Sigmoid activation function. We trained our model with a batch size of 32 using Adam as optimizer (Kingma, 2014), following the reference paper, with a standard learning rate of 0.001. The loss function employed is the Categorical Cross-Entropy, which is commonly used for multi-class classification problems.

The training in our experiment, followed two different setups:

- In the final comparison between the CL algorithms and baselines, we rigidly adhered to the procedure outlined in the cited paper. That is, we used a split of 70% training samples and 30% test samples, training the model for a total of 300 epochs. To avoid overfitting we adopted dropout layers in the model structure as well as using early stopping during training.
- To tune the hyperparameters for all CL methods, we sampled half of the test data as validation for early stopping, which was crucial for obtaining results in a reasonable amount of time. A single Nvidia Titan V GPU was used to ensure comparability of the results. With this strategy, many training sessions concluded around the 50th or 100th epoch while still providing reliable indicators for the optimal parameters.

Due to varying data points among drivers in the dataset, the number of windows per class was inconsistent (see Table 1), which could affect results when changing class order in tasks. To reduce randomness from class ordering and sampling, we ran each experiment with 4 different seeds and 4 class order permutations.

4.3. Continual learning approaches

Here, we describe the specific hyperparameters considered for each CL approach, their purpose, and the value used in the final comparison.

Table 1
Count of windows in training and test set for each driver's drive session.

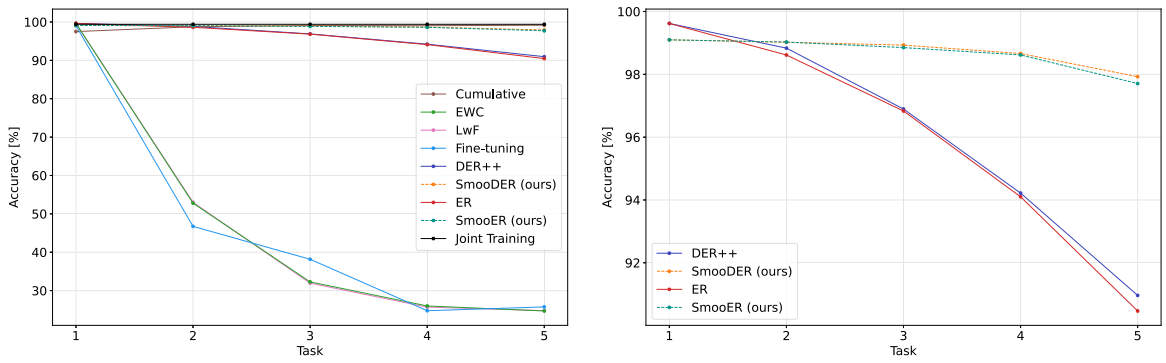
Driver ID	Session	Train windows (70%)	Test windows (30%)
0	1	367	156
0	2	466	199
1	1	773	330
1	2	717	306
2	1	455	194
2	2	409	174
3	1	701	300
3	2	831	356
4	1	509	217
4	2	463	198
5	1	677	298
5	2	596	255
6	1	404	173
6	2	458	195
7	1	567	242
7	2	574	245
8	1	427	182
8	2	472	201
9	1	417	178
9	2	610	261

Unless otherwise stated, all methods use the learning rate, optimizer, and batch size values outlined in the previous section. In Appendix A, we provide a detailed explanation of the hyperparameters tuning process, including the reasoning behind the selection of each optimal value.

- **ER**: This method needs two hyperparameters. The first is *Memory Size*, which is also the most important as it is directly related to the resource constraints. The optimal value of **1000** was selected for this parameter, as it limits memory usage without causing a significant performance loss compared to the Joint Training baseline. The second hyperparameter is then *Replay Ratio*, it essentially governs the mixing ratios of instances from the training set and the memory in a batch of fixed size. Its impact is marginally on the performance and mainly on the time required for training, as the training set is diluted into more and more iterations when the ratio of memory instances increases. Here we chose to mix the two in equal parts, so its final value is **0.5**.
- **EWC and LwF**: They both have a single hyperparameter λ that is used as a weight to balance the influence of their regularization loss before adding it to the training loss. The best values that resulted from our experiments are **10000** for EWC and **5** for LwF.
- **DER++**: This method includes both rehearsal and regularization components. Similar to ER, the more important hyperparameter is the rehearsal *Memory Size*. Despite needing slightly more resources than ER to store additional logits, we chose to keep the optimal value at **1000**. The optimal values for the parameters α and β are both **1**, which respectively balance the importance of the logits and ground truth labels extracted from the memory.
- **SmooDER and SmooER**: The hyperparameters for these methods remain the same as in their original versions, as the smoothing process is independent of the underlying approach. An additional parameter, the *smoothing window size*, is set to six, based on the dataset and data processing characteristics. This choice is aimed at the maximum allowed size to effectively reduce errors while limiting the worst-case delay during a driver change. Given the sampling frequency of 1 Hz and a data windowing stride that can be set to one sample during deployment, this choice results in a maximum delay of six seconds.

4.4. Evaluation metrics

In the following section, we present a concise overview of the metrics used in Section 5 to assess the performance of all CL approaches



(a) Test accuracy of each method and baseline.

(b) Focus on the best methods.

Fig. 5. Results for Scenario 1.

Table 2
Performance for each scenario and for each CL technique. The best accuracy for each scenario is highlighted in bold.

	Scenario 1 Two new drivers		Scenario 2 One new driver		Scenario 3 Two new sessions		Average of All scenarios	
	ACC ↑	gap ↓	ACC ↑	gap ↓	ACC ↑	gap ↓	ACC ↑	gap ↓
Cumulative	99.16	0.24	98.96	0.44	97.43	1.97	98.52	0.88
Joint Training	99.40	–	99.40	–	99.40	–	99.40	–
Fine-tuning	24.55	74.85	10.87	88.53	19.17	80.23	18.20	81.20
ER	90.47	8.93	86.25	13.15	87.08	12.32	87.93	11.47
EWC	24.69	74.71	10.92	88.48	19.10	80.30	18.24	81.16
LwF	24.78	74.62	10.88	88.52	20.01	79.39	18.56	80.84
DER++	90.96	8.44	87.04	12.36	88.87	10.53	88.96	10.44
SmooDER (ours)	**97.93**	**1.47**	**95.47**	**3.93**	**98.22**	**1.18**	**97.21**	**2.19**
SmooER (ours)	97.71	1.69	95.26	4.14	97.45	1.95	96.81	2.59

across the three scenarios. A more detailed explanation, including the mathematical formulas, can be found in Appendix C.

We define with ACC the accuracy calculated on the entire dataset after the model is trained on the last task of the data stream. Moreover, we report also the *gap*, which measures the difference between the accuracy obtained by the Joint Training approach (upper bound) and the ACC value of the CL approaches.

5. Results

In the following we present the results for each scenario, highlighting the performance of all tested CL methods and comparing these approaches with our SmooER and SmooDER methods.

In Section 5.1, we discuss results for the *Two New Drivers* scenario, followed by the *One New Driver* scenario in Section 5.2, and the *Two New Sessions* scenario in Section 5.3. We provide two figures for each scenario, with the left one comparing all the methods and baselines mentioned and the right one focusing on only the best non-baseline methods. A summary table presents the performance of each method in each scenario after learning the last task. We then conclude with an overall analysis of the conducted experiments and results in Section 5.4.

5.1. Scenario 1 - two new drivers

This scenario has the least complexity among the three considered due to its shorter sequence length. As sequence length increases, the effects of Catastrophic Forgetting are expected to become more pronounced. In Table 2 this resulted in the final accuracy for this scenario being the highest among the three. Corresponding to the classic Split-CIFAR10, its role is to link the CL literature with our work before adding the additional challenges in Scenarios 2 and 3.

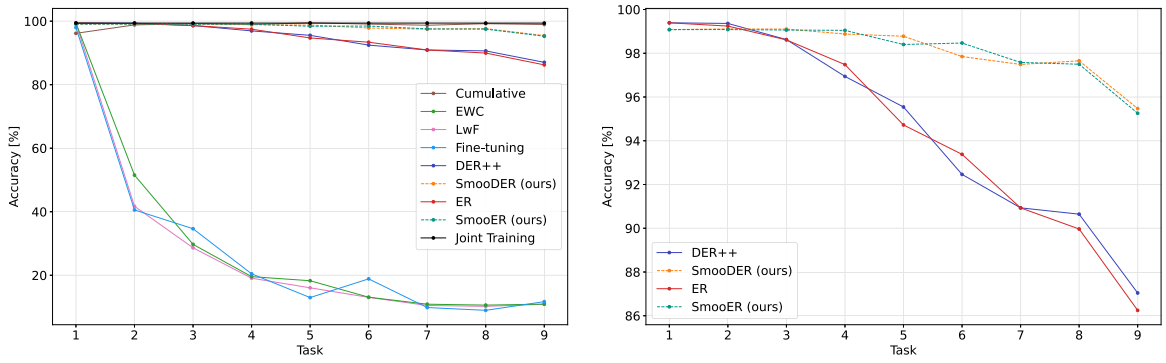
From the evolution of the accuracy along the sequence of tasks reported in Fig. 5(a), we can distinguish three groups of methods: purely regularization-based methods like LwF and EWC have no effect in contrasting forgetting in this context as their evolution is very similar to Fine-Tuning. ER and DER++ instead demonstrate significantly better accuracy, where, as expected, DER++ outperformed ER since it is a more advanced method. Specifically, ER achieved 90.47% accuracy, while DER++ 90.96%. Compared the upper bound of 99.4% set by Joint Training (Table 2), this translates in a gap of less than 9% for DER++.

Lastly, our proposed methods, SmooDER and SmooER, further enhance resistance to forgetting, achieving approximately 98% accuracy, with a 7% accuracy gain and narrowing the gap to less than 2%. Fig. 5(b) highlights the comparison between our solutions, ER, and DER++, demonstrating their clear advantages, even with a small number of tasks.

5.2. Scenario 2 - one new driver

As described in Section 3.2, considering a single new driver for each task to increase the realism resulted in a more challenging scenario as we are now dealing with twice the number of tasks. Table 2 shows that, as expected, the performance of all methods declined compared to Scenario 1, with the best-performing ones, ER and DER++, experiencing a reduction of around 4% in accuracy. Notably, SmooDER and SmooER demonstrated even lower degradation, with a decrease of only about 2.5%. Given the longer sequence and the complexity of learning each driver individually, this minor drop indicates that our methods have an acceptably low level of performance degradation as the number of tasks increases.

Fig. 6(a) presents an accuracy progression very similar to Scenario 1, with a notable observation: without effective countermeasures against forgetting, accuracy can drop as low as 11%, as seen with LwF, EWC, and Fine-Tuning. In this more complex scenario, the accuracy gap between Joint Training and DER++ (similarly ER) has increased to about 13%. While this gap might still be considered acceptable, it highlights the need for further improvement to mitigate forgetting



(a) Test accuracy of each method and baseline.

(b) Focus on the best methods.

Fig. 6. Results for Scenario 2.

and enhance performance. In this regard, our methods, SmooDER and SmooER, achieve a final accuracy of approximately 95%, reducing the gap to Joint Training (99.4%) to less than 5%. This translates to an improvement of roughly 8% over ER and DER++ which can be better appreciated from Fig. 6(b), demonstrating the robustness of our approach in managing forgetting in more challenging scenarios.

5.3. Scenario 3 - two new sessions

In Scenario 3, complexity is further increased by including partial data from two classes per task, resulting in a sequence of length ten. Here, each task may involve learning a new driver or expanding knowledge of an already encountered one. Despite this added complexity, the performance of various CL methods shows a slight improvement over Scenario 2. This observation provides two key insights: first, it suggests that the primary challenge still lies in the length of the task sequence. Second, the slight performance boost hints that learning multiple drivers per task enhances the model's capabilities to learn how to effectively distinguish them.

Fig. 7(a) presents a different accuracy progression compared to the other scenarios. This difference arises because the test set includes both driving sessions' test data as soon as each driver's first training session is introduced (see Fig. 1). As a result, accuracy tends to improve in the later tasks as the model learns the final session for each driver. This highlights the effectiveness of methods able to counteract forgetting in this context, as they display a U-shaped accuracy trend, unlike methods such as LwF and EWC.

Fig. 7(b) focuses specifically on the best methods. ER and DER++ achieve accuracy levels of 87% and 89%, respectively, maintaining a gap of around 10% from Joint Training. Notably, our methods, SmooER and SmooDER, show an even better accuracy, reaching 97.45% and 98.22%, respectively, thus narrowing the gap to the upper bound to approximately 2%. As a final consideration, the random inclusion of domain-shift elements—when the second driving session is introduced for each driver—likely contributes to the approximately 10% performance improvement of methods like EWC and LwF compared to Scenario 2.

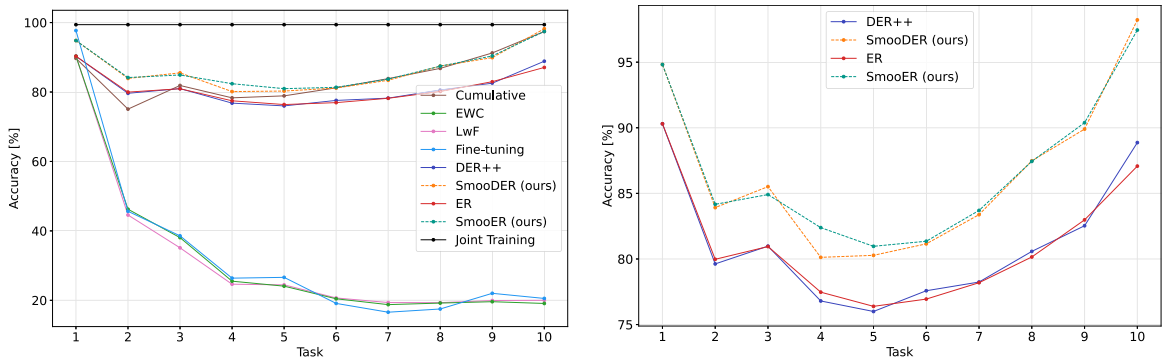
5.4. Overall analysis

In summary, SmooDER consistently outperforms all other methods across all three scenarios. The average results presented in Table 2 show that DER++ and ER achieve average accuracies of 88.96% and 87.93%, respectively, making them more effective than LwF and EWC. The latter methods yield significantly lower accuracies of 18.24% and 18.56%, only marginally surpassing the Fine-Tuning baseline at 18.20%. This can be attributed to their design being less suited to our context, which involves frequent introductions of new classes. Specifically, EWC is more effective at addressing domain shifts than at handling challenges related to class expansion, while LwF relies on using current task samples alongside guidance from a model trained on earlier tasks. This approach, however, does not efficiently recover information in tasks where, for example, all instances represent newly added drivers, as the teacher model becomes ineffective in such cases.

In contrast, methods that incorporate rehearsal, such as DER++ and ER, use a small yet representative subset of previously seen classes during each task, enabling the model to learn all seen classes jointly. This approach not only prevents forgetting but also enhances learning efficiency. Although these approaches perform well, a measurable gap (10.44%) remains compared to the Joint Training reference, where all data is trained simultaneously. SmooDER and SmooER address this challenge more effectively, reducing the gap to just 2.19% and demonstrating their capability to achieve near-optimal accuracy.

Scalability in terms of both memory and time requirements is another key result, which we have not addressed in previous sections. Fig. 8 illustrates the training time (in seconds) for each task across all CL methods in Scenario 3. Among these, the Cumulative baseline, currently used for adding new users, achieves optimal accuracy by retraining from scratch with both old and new data. However, this approach leads to linearly increasing time requirements per each new task. In contrast, Fine-Tuning only processes new data, resulting in nearly constant training time across tasks. LwF, positioned between these approaches, also maintains constant time requirements, as it only computes a regularization term without significant extra computation. Meanwhile, EWC shows a growing time curve, as its implementation requires a separate weight matrix for each task. Alternative versions of EWC, which condense these matrices, are available to reduce training time. Memory-based methods like DER++ and ER offer a constant time curve, situated between Fine-Tuning and Cumulative, depending on memory size. It is important to note that CL methods do not take effect during the first task, so these considerations apply from task 2 onward. Lastly, SmooDER and SmooER require training and testing times similar to DER++ and ER respectively, as the additional computational overhead is minimal.

Finally, our approach has intrinsic modularity, where the underlying model, the CL method, and the prediction smoothing technique do not impose constraints on each other. Therefore, any future advancement on any component can be easily integrated into the system. This makes it possible to extend our work to other fields, as long as the following properties hold. The input should be a time series, a good underlying model for that problem needs to be available, and the labels are generally consistent in time. In human activity recognition, for example, we can employ this approach to enable the training of the model on user-defined activities. When dealing with faults, which are inherently persistent over time, our approach allows for the definition of faults specific to the structure of a particular machine, a characteristic often seen with hydroelectric turbines.



(a) Test accuracy of each method and baseline.

(b) Focus on the best methods.

Fig. 7. Results for Scenario 3.

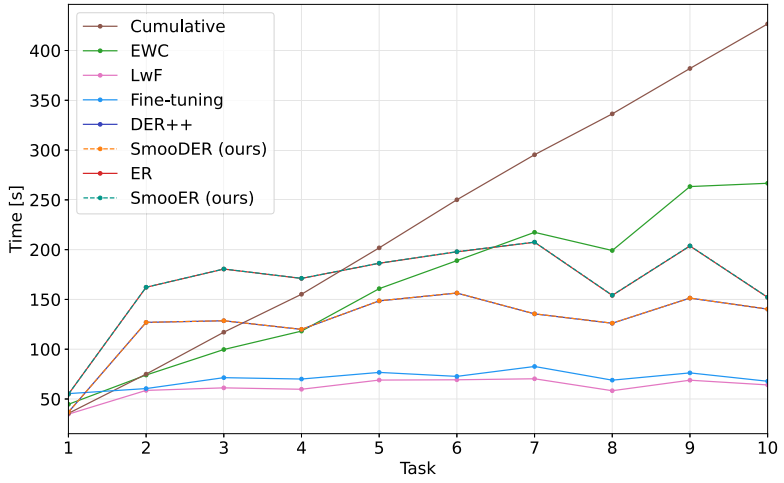


Fig. 8. Training time per task for each method and baseline in Scenario 3. Cumulative training time increases linearly as more tasks are added since the datasets are concatenated. Fine-tuning and LwF have negligible or no computational overhead. EWC computational time grows since by implementation details each task contributes independently to the regularization. DER++, ER, SmooDER, and SmooER go to saturation when the memory is filled for the first time. The overhead of SmooDER over DER++ is negligible (same for SmooER over ER).

The results of our work show the superiority of our approach. This is due to the purposely tough challenges we put in our scenarios, with many tasks and the rapid introduction of new classes.

Regarding memory requirements, Cumulative needs as training dataset the concatenation of all data from the previous and the current task, so the memory requirement grows linearly in the number of tasks, amounting to a total of 120 MB in our experiments. Instead, CL approaches only use the data from the current task so the memory requirement remains constant. Moreover, the highest-performing methods DER++ and ER (as well as their respective variants, SmooDER and SmooER, which are identical in memory requirements) only need to store a small additional subset of samples for rehearsal.

In the results shown in Table 2, we use 1000 samples, occupying just 11 MB — specifically, less than $1000 \times (60 \times 46 \times 4B + 8B + 10 \times 4B) = 11,088,000B$ for DER++ and $1000 \times (60 \times 46 \times 4B + 8B) = 11,048,000B$ for ER. Including the network size of 0.89 MB, a total of only 12 MB of memory is needed. Finally, for the deployment environment, just 6 MB of RAM is sufficient: 0.89 MB for the network, 3.93 MB for the backward pass during training, and 0.35 MB for the input batch. This efficient memory usage enables direct deployment on small devices, such as those commonly installed in vehicles (A.G., 2024).

When considering the eventual implementation of this system, it is important to note that during training, the CL formulation dataset size remains constant, whereas the Cumulative dataset grows linearly as all collected data is concatenated. The best-performing CL methods, however, require only an additional 11MB of non-volatile memory to store the samples for rehearsal. During inference, CL methods do not impact performance, so the one-second latency is mainly driven by the 1 Hz sampling frequency needed to generate a new data window. SmooDER and SmooER may hold the predicted driver ID for up to six seconds, but only during a driver change.

6. Conclusion

In this work, we addressed the challenge of Driver Identification. We moved beyond the traditional approach, where all driver data is collected before model deployment, and training occurs offline. In many real-world scenarios, like car-sharing services where new drivers are continuously added as they request to access the vehicle, the traditional complete retraining policy is inadequate due to limited computational resources and memory. On the other hand, training only on new data is impractical because the model would lose the ability to recognize already authorized users, a phenomenon known as Catastrophic Forgetting. Continual Learning allows the system to not only add new drivers as they come but also to incorporate new data from previously seen drivers. This approach enables a system that progressively adapts and enhances its performance as new data is collected.

The main goal of this work was to apply well-established CL approaches, such as EWC, LwF, ER, and DER++, to an existing optimal solution for static Driver Identification and evaluate their effectiveness in reducing Forgetting compared to Fine-Tuning, as well as limiting

the resource requirements compared to Cumulative Training when transitioning to the CL formulation. To test our hypothesis, we designed three progressively realistic and complex scenarios. The implementation considers the widely known OCSLab dataset. Additionally, we took into account the temporal correlation among samples during the inference phase and introduced two new methods, SmooDER and SmooER, which are designed to leverage this information to further minimize Forgetting.

The average accuracy of our baselines across all three scenarios is: 99.40% and 98.52% for the upper bounds (Joint Training and Cumulative Training, respectively) and 18.20% for Fine-Tuning. In contrast, the accuracy for the CL methods is as follows: 88.96% for DER++, 87.93% for ER, 18.56% for LwF, and 18.24% for EWC. These results demonstrate that even in challenging environments like our scenarios, ER and DER++ are effective solutions, as they managed to reduce the linear growth of the required time for each new task of the Cumulative Training approach to an approximately constant time, while only losing an average of about 11% accuracy from Joint Training due to forgetting. Furthermore, our methods, SmooDER and SmooER, proved to be the best approaches, reducing the gap to 2.19% and 2.59%, respectively, with negligible computational overhead compared to DER++ and ER.

While our work is extensive in considering multiple aspects of the problem, behavior-based driver identification is a complex field and the are still many opportunities for future research. For example, although the smoothing technique is highly effective, the proposed approach is relatively simple. Incorporating a more advanced method, such as an adaptive window size or a dynamic smoothing technique, could further minimize misclassifications.

Moreover, there is significant potential for future research into the effects of adversarial and data poisoning attacks on behavior-based driver identification systems that use continual learning. Our system does not introduce any specific vulnerabilities that are easily exploitable. All components are integrated within the microcontroller, and physical access would be required to alter them. However, we assume the system inherits the vulnerabilities of the underlying model. Consequently, adversarial attacks may lead to misclassifications, and data poisoning could degrade model performance over time. Therefore, future work should focus on investigating these aspects and on developing defenses that balance adaptability and robustness.

Finally, the modularity and flexibility of our approach demonstrated potential for future application in other domains with similar properties, such as human activity recognition, industrial fault detection, or healthcare.

CRediT authorship contribution statement

Mattia Fanan: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation, Formal analysis, Data curation. **Davide Dalle Pezze:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Investigation, Conceptualization. **Emad Efatinasab:** Validation, Methodology, Data curation, Conceptualization. **Ruggero Carli:** Supervision, Resources, Project administration. **Mirco Rampazzo:** Resources, Project administration. **Gian Antonio Susto:** Writing – original draft, Resources, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2025.110459.

Data availability

Public dataset.

References

Abu-Gellban, H., Nguyen, L., Moghadasi, M., Pan, Z., Jin, F., 2020. Livedi: An anti-theft model based on driving behavior. In: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security. pp. 67–72.

A.G., I.T., 2024. AURIX™ Family – TC39xXX. https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc3xx/aurix-family-tc39xxx/ (Accessed 05 October 2024).

Ahmadi-Assalemi, G., Al-Khateeb, H.M., Maple, C., Epiphaniou, G., Hammoudeh, M., Jahankhani, H., Pillai, P., 2021. Optimising driver profiling through behaviour modelling of in-car sensor and global positioning system data. Comput. Electr. Eng. 91, 107047.

Azadani, M.N., Boukerche, A., 2020. Performance evaluation of driving behavior identification models through can-bus data. In: 2020 IEEE Wireless Communications and Networking Conference. WCNC, IEEE, pp. 1–6.

Azadani, M.N., Boukerche, A., 2021. Driver identification using vehicular sensing data: A deep learning approach. In: 2021 IEEE Wireless Communications and Networking Conference. WCNC, IEEE, pp. 1–6.

Bugarin, N., Bugaric, J., Barusco, M., Pezze, D.D., Susto, G.A., 2024. Unveiling the anomalies in an ever-changing world: A benchmark for pixel-level anomaly detection in continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4065–4074.

Burton, A., Parikh, T., Mascarenhas, S., Zhang, J., Voris, J., Artan, N.S., Li, W., 2016. Driver identification and authentication with active behavior modeling. In: 2016 12th International Conference on Network and Service Management. CNSM, IEEE, pp. 388–393.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S., 2020. Dark experience for general continual learning: a strong, simple baseline. arXiv:2004.07211 URL https://arxiv.org/abs/2004.07211.

Ceccon, M., Pezze, D.D., Fabris, A., Susto, G.A., 2024. Multi-label continual learning for the medical domain: A novel benchmark. arXiv preprint arXiv:2404.06859.

Chen, J., Wu, Z., Zhang, J., 2019. Driver identification based on hidden feature extraction by using adaptive nonnegativity-constrained autoencoder. Appl. Soft Comput. 74, 1–9.

Cura, A.h., Küçük, H., Ergen, E., Öksüzoğlu, I.B., 2021. Driver Profiling Using Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN) Methods. IEEE Trans. Intell. Transp. Syst. 22 (10), 6572–6582. http://dx.doi.org/10.1109/TITS.2020.2995722, Conference Name: IEEE Transactions on Intelligent Transportation Systems.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T., 2021. A continual learning survey: Defying forgetting in classification tasks. IEEE Trans. Pattern Anal. Mach. Intell. 44 (7), 3366–3385.

van de Ven, G.M., Tolias, A.S., 2019. Three scenarios for continual learning. CoRR abs/1904.07734 arXiv:1904.07734 URL http://arxiv.org/abs/1904.07734.

Efatinasab, E., Donadel, D., Conti, M., 2022. GAN-CAN: A Novel Attack to Behavior-Based Driver Authentication Systems. (Master Thesis in Computer Science, Department of Mathematics). "Tullio Levi-Civita. University of Padova, https://hdl.handle.net/20.500.12608/52324.

Efatinasab, E., Marchiori, F., Donadel, D., Brighente, A., Conti, M., 2024. When authentication is not enough: On the security of behavioral-based driver authentication systems. arXiv preprint arXiv:2306.05923 arXiv:2306.05923.

El Mekki, A., Bouhoute, A., Berrada, I., 2019. Improving driver identification for the next-generation of in-vehicle software systems. IEEE Trans. Veh. Technol. 68 (8), 7406–7415.

Erzin, E., Yemez, Y., Tekalp, A.M., Erçil, A., Erdogan, H., Abut, H., 2006. Multimodal person recognition for human-vehicle interaction. IEEE MultiMedia 13 (2), 18–31.

Ezzini, S., Berrada, I., Ghogho, M., 2018. Who is behind the wheel? Driver identification and fingerprinting. J. Big Data 5 (1), 1–15.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D., 2017. Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734.

Francillon, A., Danev, B., Capkun, S., 2011. Relay attacks on passive keyless entry and start systems in modern cars. In: Proceedings of the Network and Distributed System Security Symposium. NDSS, Eidgenössische Technische Hochschule Zürich, Department of Computer Science.

Gahr, B., Liu, S., Koch, K., Barata, F., Dahlinger, A., Ryder, B., Fleisch, E., Wortmann, F., 2019. Driver identification via the steering wheel. arXiv preprint arXiv:1909.03953.

Garcia, F.D., Oswald, D.F., Kasper, T., Pavlidès, P., 2016. Lock it and still lose it-on the (in) security of automotive remote keyless entry systems.. In: USENIX Security Symposium. Vol. 53.

Girma, A., Yan, X., Homaifar, A., 2019. Driver identification based on vehicle telematics data using lstm-recurrent neural network. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence. ICTAI, IEEE, pp. 894–902.

Hallac, D., Sharang, A., Stahlmann, R., Lamprecht, A., Huber, M., Roehder, M., Leskovec, J., et al., 2016. Driver identification using automobile sensor data from a single turn. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems. ITSC, IEEE, pp. 953–958.

Hill, K., 2024. Automakers Are Sharing Consumers' Driving Behavior With Insurance Companies. The New York Times, http://www.nytimes.com/2024/03/11/technology/carmakers-driver-tracking-insurance.html (Accessed 20 March 2024).

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hochreiter, S., 1997. Long Short-term Memory. Neural Computation MIT-Press.

Kang, Y.G., Park, K.H., Kim, H.K., 2019. Automobile theft detection by clustering owner driver data. arXiv preprint arXiv:1909.08929.

Kingma, D.P., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., 2017. Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. 114 (13), 3521–3526.

Kwak, B.I., Woo, J., Kim, H.K., 2016. Know your master: Driver profiling-based anti-theft method. In: 2016 14th Annual Conference on Privacy, Security and Trust. PST, IEEE, pp. 211–218.

Li, Z., Hoiem, D., 2017. Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. 40 (12), 2935–2947.

Lin, X., Zhang, K., Cao, W., Zhang, L., 2018. Driver evaluation and identification based on driving behavior data. In: 2018 5th International Conference on Information Science and Control Engineering. ICISCE, IEEE, pp. 718–722.

Lomonaco, V., Maltoni, D., 2017. Core50: a new dataset and benchmark for continuous object recognition. In: Conference on Robot Learning. PMLR, pp. 17–26.

Lopez-Paz, D., Ranzato, M., 2017. Gradient episodic memory for continual learning. Adv. Neural Inf. Process. Syst. 30.

Mallya, A., Lazebnik, S., 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7765–7773.

Marchegiani, L., Posner, I., 2018. Long-term driving behaviour modelling for driver identification. In: 2018 21st International Conference on Intelligent Transportation Systems. ITSC, IEEE, pp. 913–919.

Martinelli, F., Mercaldo, F., Orlando, A., Nardone, V., Santone, A., Sangaiah, A.K., 2020. Human behavior characterization for driving style recognition in vehicle system. Comput. Electr. Eng. 83, 102504.

McCloskey, M., Cohen, N.J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of Learning and Motivation. Vol. 24, Elsevier, pp. 109–165.

Merlin, G., Lomonaco, V., Cossu, A., Carta, A., Bacciu, D., 2022. Practical recommendations for replay-based continual learning methods. In: International Conference on Image Analysis and Processing. Springer, pp. 548–559.

Pasti, F., De Monte, R., Pezze, D.D., Susto, G.A., Bellotto, N., 2024. Tiny robotics dataset and benchmark for continual object detection. arXiv preprint arXiv:2409.16215.

Qiao, Z., Pham, X.H., Ramasamy, S., Jiang, X., Kayacan, E., Sarabakha, A., 2024. Continual learning for robust gate detection under dynamic lighting in autonomous drone racing. In: 2024 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Rahim, M.A., Zhu, L., Li, X., Liu, J., Zhang, Z., Qin, Z., Khan, S., Gai, K., 2019. Zero-to-stable driver identification: A non-intrusive and scalable driver identification scheme. IEEE Trans. Veh. Technol. 69 (1), 163–171.

Ravi, C., Tigga, A., Reddy, G.T., Hakak, S., Alazab, M., 2022. Driver identification using optimized deep learning model in smart transportation. ACM Trans. Internet Technol. 22 (4), 1–17.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G., 2019. Experience replay for continual learning. Adv. Neural Inf. Process. Syst. 32.

Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R., 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671.

Shieh, J.-L., Haq, Q.M.u., Haq, M.A., Karam, S., Chondro, P., Gao, D.-Q., Ruan, S.-J., 2020. Continual learning strategy in one-stage object detection framework based on experience replay for autonomous driving vehicle. Sensors 20 (23), 6777.

Singh, A., Gurbuz, M.B., Gantha, S.S., Jasti, P., 2023. Class-incremental continual learning for general purpose healthcare models. arXiv preprint arXiv:2311.04301.

Soutif-Cormerais, A., Carta, A., Cossu, A., Hurtado, J., Hemati, H., Lomonaco, V., de Weijer, J.V., 2023. A comprehensive empirical evaluation on online continual learning. arXiv:2308.10328 URL https://arxiv.org/abs/2308.10328.

Tiwari, R., Killamsetty, K., Iyer, R., Shenoy, P., 2022. Gcr: Gradient coreset based replay buffer selection for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 99–108.

Tseng, P.-Y., Lin, P.-C., Kristianto, E., 2023. Vehicle theft detection by generative adversarial networks on driving behavior. Eng. Appl. Artif. Intell. 117, 105571. http://dx.doi.org/10.1016/j.engappai.2022.105571, URL https://www.sciencedirect.com/science/article/pii/S0952197622005619.

Verwimp, E., Yang, K., Parisot, S., Hong, L., McDonagh, S., Pérez-Pellitero, E., De Lange, M., Tuytelaars, T., 2023. Clad: a realistic continual learning benchmark for autonomous driving. Neural Netw. 161, 659–669.

Xie, G., Wang, J., Liu, J., Lyu, J., Liu, Y., Wang, C., Zheng, F., Jin, Y., 2024. Im-iad: industrial image anomaly detection benchmark in manufacturing. IEEE Trans. Cybern..

Xun, Y., Liu, J., Kato, N., Fang, Y., Zhang, Y., 2019. Automobile driver fingerprinting: A new machine learning based authentication scheme. IEEE Trans. Ind. Informatics 16 (2), 1417–1426.

Xun, Y., Liu, J., Kato, N., Fang, Y., Zhang, Y., 2020. Automobile driver fingerprinting: A new machine learning based authentication scheme. 16, (2), pp. 1417–1426. http://dx.doi.org/10.1109/TII.2019.2946626,

Zhang, J., Wu, Z., Li, F., Xie, C., Ren, T., Chen, J., Liu, L., 2019. A deep learning framework for driving behavior identification on in-vehicle CAN-bUS sensor data. Sensors 19 (6), 1356.