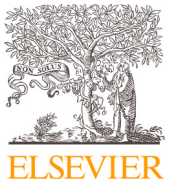
Transportation Research Part C 190 (2026) 105763



Contents lists available at ScienceDirect

Transportation Research Part C

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

Constrained machine unlearning of learning-based intelligent transportation systems models

Xin Wang [a], R. Tyrrell Rockafellar [b], Xuegang (Jeff) Ban [a,1,*]

[a] Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, 98195, United States
[b] Department of Mathematics, University of Washington, Seattle, WA, 98195, United States

ARTICLE INFO

Keywords:
Machine unlearning
Constrained optimization
Privacy
Intelligent transportation systems (ITS)

ABSTRACT

Learning-based Intelligent Transportation Systems (ITS) models rely heavily on data sources that may contain sensitive or poisoned information. While the abundance of data has fueled significant breakthroughs, particularly in data-driven, machine learning based ITS methods, it also raises concerns regarding privacy, cybersecurity, and data freshness. These issues can erode public trust in ITS. Recently, regulations have introduced the "right to be forgotten", allowing users to request the removal of their private data from models. As machine learning models can remember old data, simply removing the data from back-end databases is insufficient in such systems.

To address these challenges, this paper introduces *constrained machine unlearning* for ITS learning models, which enables a trained ITS model to selectively forget privacy-sensitive, poisoned, or outdated data, without retraining. The proposed unlearning framework is based on the sensitivity analysis of constrained learning problems and is validated using machine learning and deep learning-based ITS applications. By empowering models to "unlearn," we aim to enhance the trustworthiness and reliability of data-driven ITS.

1. Introduction

Learning-based methods and Intelligent Transportation Systems (ITS) have co-evolved over recent years. Advances in machine learning (ML) and deep learning (DL) provide increasingly powerful tools for ITS modeling, prediction, and control, while ITS continuously generates large volumes of data and application-driven challenges that motivate new ML/DL methodologies (Veres and Moussa, 2019; Haydari and Yılmaz, 2020). Along this trajectory, learning-based methods in ITS have evolved from traditional ML methods to convolutional and recurrent neural networks (CNNs and RNNs), and graph-based architectures (GCNs), and more recently to large language model (LLM) frameworks. As ML/DL models continue to scale up, they can process multimodal data, including images, videos, GPS trajectories, and infrastructure-based measurements (e.g., loop detector data). At the same time, the growing complexity of these models substantially increases the volume of data required for training. As a result, ML/DL-based ITS is increasingly dependent on access to large-scale, heterogeneous traffic data collected from diverse sources. These data sources can be broadly categorized into infrastructure-collected data and user-contributed data. Infrastructure-collected data typically includes information collected from loop detectors, traffic cameras, and radars installed on roadways or at intersections. In contrast, user-contributed data is derived from individuals, often through vehicles or personal devices, such as GPS traces, vehicle trajectories, and probe data collected via mobile apps or in-vehicle systems. Additionally, V2X communication data (Hasan et al., 2020), which bridges these two categories, involves information exchanged among vehicles, other road users, and infrastructure, or directly between vehicles.

* Corresponding author.
E-mail address: banx@uw.edu (X. Ban).
[1] Editor.

https://doi.org/10.1016/j.trc.2026.105763
Received 22 December 2025; Received in revised form 14 April 2026; Accepted 12 May 2026

Although the integration of diverse data has significantly improved the capacity and reliability of ITS, it also raises concerns about legal, privacy, and safety violations stemming from adversarial data manipulation (e.g., data poisoning attacks), datasets containing privacy sensitive information, and incorrect or outdated training data (Wang et al., 2024b; De Montjoye et al., 2013). For example, GPS traces or trajectory data linked to specific vehicles could accidentally expose sensitive locations of individuals, posing significant privacy and security risks to the vehicle owners (Iqbal and Lim, 2010). In addition, outdated or poisoned training data can lead to inaccurate traffic state estimation or predictions (Wang et al., 2024b,c), which undermines the reliability and trustworthiness of ITS models. As ML/DL models are known to potentially memorize training data, only removing data from back-end databases (without retraining the model) is often not sufficient to remove the impacts of the problematic data. A straightforward solution to these issues might involve simply removing the sensitive or poisoned data and retraining the model from scratch. However, as ML/DL models grow increasingly large and complex, retraining can be prohibitively expensive, requiring vast computational resources and time. Thus, it is impractical to retrain models every time data needs to be removed, underscoring the demand for efficient learning approaches that comply with data deletion requests. This has become more pressing recently due to regulations such as the European Union's General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), Canada's Personal Information Protection and Electronic Documents Act (PIPEDA), and China's Personal Information Protection Law (PIPL). These regulations enforce the "right to be forgotten" (Regulation, 2016), which requires organizations to delete personal data on request and to take reasonable steps to erase the impacts of such data from their systems, including ML/DL models trained using the data.

The process of deleting data and its impacts from an ML or DL model, without retraining, is known as *machine unlearning* (Bourtoule et al., 2021; Nguyen et al., 2022; Triantafillou et al., 2024). It aims to produce a modified model, referred to as the *unlearned model*, that is equivalent to, or at least behaves like, a model fully retrained on the remaining data after the removal request has been processed (which we refer to as the *gold standard model*). Noteworthy is that, due to the nature of data deletion requests, it is expected that the removed dataset is typically *small* compared with the entire training dataset. Appendix D and Appendix E provide further discussion on this aspect.

In this study, we focus on machine unlearning of ITS models that leverage ML/DL techniques. Examples include Least Squares (LS) models to estimate delay patterns using mobile data (Ban et al., 2011), Support Vector Machines (SVMs) applied to vehicle classification using vehicle trajectory data (Sun and Ban, 2013), and long short-term memory (LSTM) neural networks for traffic flow prediction (Li et al., 2020). Recently, physics-informed neural networks (PINNs) have emerged as a novel method that combines traffic domain knowledge with learning-based approaches (Shi et al., 2021; Lu et al., 2023; Thodi et al., 2022). This integration seeks to overcome the "black-box" nature of neural networks applied by data-driven methods, offering more interpretable solutions that align closely with fundamental traffic principles. Note that both classical ML/DL-based and PINN-based ITS models often incorporate problem-specific constraints. For instance, a model estimating vehicle queue lengths at intersections must consider the upper and lower limits of the signal's green time and the physical car-following behavior between vehicles (Ban et al., 2011). PINN-based traffic state estimations must ensure that their solutions adhere to traffic flow theories described by physical models, such as the Lighthill-Whitham-Richards (LWR) model (Shi et al., 2021). In this paper, we refer to the unlearning of data-driven learning models with constraints as *constrained machine unlearning* (CMU).

After receiving a data deletion request, machine unlearning can "erase" the influence of the data to be forgotten (e.g., a subset of trajectory data) from the trained ITS model, adjusting the model's solution (e.g., neural network weights) towards the solution of the gold standard model. Existing machine unlearning frameworks often assume the absence of constraints and implement the removal by directly estimating the solution change caused by data removal via gradient-based methods. Further discussion is provided later in Section 2. However, when the problem involves constraints (e.g., traffic flow conservation laws or physical car-following behavior), the solution space is restricted, and the unlearning process (i.e., CMU) must ensure that the adjusted solution remains feasible under these constraints. Furthermore, since the constraints are often data-dependent, data removal can also modify the feasible region. This requires CMU to measure the influence of the removed data on both the objective function and the constraints simultaneously to ensure that the solution of unlearned model is optimal and valid within the updated feasible region. To the best of our knowledge, there is no work that has fully explored CMU to explicitly deal with constraints.

Assume our ITS models rely on the training dataset $D = [z_1, z_2, \dots, z_N]$ to determine the optimal solution set θ. Consider a request to remove two data points z_1, z_2. We assign a weight vector $\eta = [\eta_1, \eta_2, \dots, \eta_N]$ to the data points, and define a solution mapping from η to θ: $S(\eta; D) = \{\theta \mid \theta \text{ satisfies the ITS model and constraints}\}$. This solution mapping allows us to transfer CMU to a sensitivity analysis problem with respect to η. By gradually reducing η_1 and η_2 from 1 to 0, we can diminish the influence of z_1 and z_2 on the solution set. The key lies in measuring how the solution θ evolves as η changes. We address this challenge through a Variational Inequality (VI) based sensitivity analysis by defining specific concepts, measures, and tools that can properly deal with constrained learning problems. Sections 3 and 4 will provide further discussion on this. Our main contributions are summarized as follows:

1) We formulate machine unlearning for ITS as a constrained data-weighted learning problem, where the removed data may affect both the objective function and the constraints.

2) Based on the VI characterization of the solution mapping, we derive a first-order approximation of the retrained solution under data removal, which leads to a practical CMU method, with unconstrained unlearning as a special case.

3) We reformulate the first-order approximation into a computable quadratic programming (QP) problem.

4) We demonstrate the effectiveness of the proposed CMU framework on representative ITS applications, including constrained SVM and PINN models, where existing unconstrained unlearning methods are not directly applicable.

The remainder of the paper is organized as follows. Section 2 provides a review of machine unlearning, based on a brief background on learning-based ITS models together with data privacy and security issues. Section 3 introduces the theoretical foundation of CMU by formulating it as a sensitivity analysis problem of constrained optimization. Section 4 customizes the CMU method for ITS and

proposes a numerical algorithm based on quadratic programming. Section 5 evaluates the proposed algorithm on two tasks: SVM-based vehicle classification and PINN-based velocity field reconstruction. Section 6 concludes the paper and discusses future research directions.

2. Background and Related Work

Learning-based ITS models represent a key development direction for future transportation systems, integrating advances in computer science, communication technologies, and sensing infrastructures to deliver safe, efficient, and reliable transportation services. As this paper does not aim to develop new ITS models, we do not provide a detailed review of existing modeling approaches. Interested readers are referred to Zhang et al. (2011), Veres and Moussa (2019), Xu et al. (2025) for comprehensive surveys of learning-based ITS models.

Benefiting from the widespread deployment of connected devices, such as navigation systems, Bluetooth-enabled equipment, and infrastructure-based sensing systems (e.g., loop detectors, traffic cameras, and connected traffic signal controllers), modern ITS continuously generates massive volumes of traffic data, reaching the scale of trillions of bytes (Gong et al., 2023). This unprecedented data availability has played a central role in motivating and enabling the successful deployment of data-driven ML/DL applications in ITS, including traffic state estimation and prediction, adaptive signal control, and intelligent driving systems (Sun et al., 2013; Wang and Yang, 2024; Feng et al., 2020; Ying et al., 2022; Zhou and Yang, 2024). At the same time, the extensive collection and utilization of large-scale traffic data raise growing concerns regarding data privacy and security. To address privacy concerns, privacy-preserving learning approaches have been widely studied. These privacy-preserving mechanisms can be implemented at different stages of the learning pipeline, including the training stage, data processing stage, and inference stage. At the training stage, federated learning (FL) has been regarded as a promising solution for mitigating privacy risks during data sharing. For example, Wang and Yang (2024) proposed a vertical federated learning framework to reduce the leakage of sensitive information during data collaboration between mobility service providers and municipal authorities. However, existing federated learning frameworks remain vulnerable to inference attacks, such as attribute inference attacks, and therefore cannot fully guarantee data privacy (Nasr et al., 2019; Wang et al., 2019; Truex et al., 2018). This vulnerability arises because sensitive information can still be implicitly revealed through gradients, model updates, or the training dynamics themselves (Zhu et al., 2019). For deep learning models, Abadi et al. (2016) incorporated differential privacy into the training process. The idea is to inject calibrated noise into gradients during stochastic optimization. This approach ensures that the presence or absence of any single training sample does not significantly change the model updates, thereby limiting the amount of information that the model can reveal about individual data records.

At the data processing stage, privacy-preserving methods rely on cryptographic and anonymization techniques. For example, Homomorphic Encryption is deployed to ensure data confidentiality during collaborative computations (Firdaus et al., 2025). At the inference stage, privacy-preserving methods focus on defending against privacy attacks, such as membership inference and attribute inference attacks (Rao et al., 2024). Adversarial machine learning is a representative approach that models the adversary's perspective and designs corresponding defenses. For instance, Qi et al. (2022) proposed contrastive adversarial learning in vertical federated learning to learn protected representations that retain fairness-relevant information while suppressing sensitive attributes. While these preventive paradigms effectively limit initial information leakage, they lack post-hoc data deletion capabilities, leaving deployed models vulnerable if specific privacy-sensitive records or corrupted samples must be explicitly removed later (e.g., due to privacy regulations such as GDPR). For data security concerns, various defense methods have been developed to detect and mitigate malicious data attacks (Wang et al., 2024a). For example, Wang et al. (2023) proposed infrastructure-based defense mechanisms that leverage secured data from roadside units to detect and correct GPS spoofing attacks. However, once corrupted or poisoned data are identified, existing ML/DL models typically need to be fully retrained after removing the identified data, which is computationally expensive and time-consuming.

The vulnerabilities of existing privacy-preserving mechanisms highlight the need to directly remove privacy-sensitive records to prevent information leakage. Naive retraining guarantees the complete removal of targeted data by training a new model from scratch, which is, however, computationally prohibitive for the massive datasets typical in ITS. Since the 2020s, machine unlearning has emerged as a promising paradigm for addressing concerns related to privacy, security, robustness, and bias mitigation. Rather than retraining models from scratch, machine unlearning aims to efficiently update trained models to "forget" specific data points, such as privacy-sensitive records or corrupted samples. As a result, machine unlearning addresses training-time information leakage left unresolved by privacy-preserving learning. Moreover, machine unlearning can be naturally integrated with data security defense mechanisms: once corrupted or malicious data are identified by defense methods, machine unlearning enables their rapid removal from deployed learning-based ITS models without incurring the high cost of full retraining.

Existing machine unlearning methods are mostly on *unconstrained* settings, which can be broadly categorized into data-based and model-based approaches. Data-based methods leverage data partitioning techniques to facilitate efficient retraining. A representative example is the SISA framework (Bourtoule et al., 2021), which partitions the dataset into multiple slices. Each slice is used to train an individual model, and the final prediction is obtained by aggregating the outputs of these models. When unlearning a data point, only the affected slice is retrained, significantly reducing computation. Model-based methods, on the other hand, aim to modify the model solutions directly in response to data deletion. For linear models, this can be efficiently achieved using rank-one updates of matrix inverses (Birattari et al., 1998; Cao and Yang, 2015). For deep neural networks, one common approach is based on influence functions (Law, 1986; Koh and Liang, 2017), which use Taylor expansions to approximate the impact of a removed data point on the model solution. However, influence function based methods typically require explicit access to Hessian matrices and are limited to handling the removal of a single data point. More recently, Golatkar et al. (2020) introduced a method inspired by differential privacy,

Table 1
Summary of Existing Machine Unlearning Approaches.

Category	Representative Approach	Strengths	Weaknesses
Data-based	SISA (Bourtoule et al., 2021)	• Model-agnostic • Exact unlearning • Conceptually simple	• High storage overhead • Retraining slices is expensive
Model-based	Rank-one updates (Birattari et al., 1998)	• Exact unlearning • Computationally efficient	• Strictly limited to linear models • Cannot generalize to DNNs
Model-based	Influence functions (Koh and Liang, 2017)	• Theoretically grounded • Applicable to DNNs	• Requires explicit Hessian • Limited to single-point removal
Model-based	Fisher information (Golatkar et al., 2020)	• Avoids explicit full Hessian • Effective for DNNs	• Approximate unlearning only • Performance degrades under sequential deletions
Optimization-based	DeltaGrad (Wu et al., 2020)	• Accelerates SGD retraining • Reuses historical gradient information	• High memory footprint • Requires hyperparameter tuning

which leverages the Fisher Information Matrix to effectively erase the influence of deleted training samples. Optimization-based approaches, such as DeltaGrad (Wu et al., 2020), accelerate the retraining process by reusing and adjusting previously computed gradients. Since model retraining typically relies on gradient descent, computing accurate gradients is the most time-consuming step. They evaluate how small changes in the dataset (e.g., the removal of a few data points) influence the gradients, and leverage this information to approximate the updated gradients without recomputing them from scratch. A summary of the aforementioned unlearning methods is provided in Table 1, with the strengths and weaknesses of each method highlighted.

Existing machine unlearning methods mostly focus on unconstrained learning tasks, and the primary mathematical tool for the analysis is *gradient* of the model solution with respect to data changes. However, for ML/DL-based ITS models with constraints, existing gradient-based methods are not applicable. This is because in the constrained setting, the change of model solution with data may not be differentiable. This paper proposes CMU, together with more advanced mathematical tools based on VI and directional derivatives, which extend existing methods to constrained optimization settings and enable the simultaneous removal of multiple data points.

3. Constrained Machine Unlearning as Sensitivity Analysis

To study CMU in the presence of constraints, we begin with a general constrained learning formulation:

$$\begin{aligned} \bar{\theta} = \arg\min_{\theta} \quad & g_0(D,\theta) \\ \text{s.t.} \quad & g_j(D,\theta) \le 0, \quad j = 1,\dots,J, \\ & h_t(D,\theta) = 0, \quad t = 1,\dots,T. \end{aligned} \tag{1}$$

Here, $D = [z_i]_{i=1}^N$ denotes a dataset, and $g_0(\cdot)$ represents the objective function. The constraints g_i and h_t represent general inequality and equality constraints, respectively. In specific learning problems, they often represent application-specific requirements and constraints. For example, as shown in Section 4, ITS problems may embed physical consistency, resource limits, or specific requirements such as fairness. $\bar{\theta}$ denotes the optimal solution derived from the full dataset. We assume that all functions are twice differentiable.

CMU aims to remove a (small) portion of the data, denoted as D^r, from the full dataset D. We denote the solution derived from the reduced dataset $D \setminus D^r$ as $\bar{\theta}_{-D^r}$:

$$\begin{aligned} \bar{\theta}_{-D^r} = \arg\min_{\theta} \quad & g_0(D \setminus D^r,\theta) \\ \text{s.t.} \quad & g_j(D \setminus D^r,\theta) \le 0, \quad j = 1,\dots,J, \\ & h_t(D \setminus D^r,\theta) = 0, \quad t = 1,\dots,T. \end{aligned} \tag{2}$$

We refer to the above problem as the *gold standard model*, as it provides the ground truth solution after data removal. As aforementioned, the gold standard model implies retraining of the original model after data is removed, which may be computationally expensive. In the following, we demonstrate how to estimate $\bar{\theta}_{-D^r}$ using $\bar{\theta}$ without retraining.

In the constrained learning (1), each data point z_i is assigned an equal weight. To remove the data points D^r, we borrow the concept of the influence function from robust statistics (Law, 1986), which was originally designed for unconstrained problems. Specifically, we introduce a data weight vector $\eta = [\eta_1, \eta_2, \dots, \eta_N] \in \mathbf{R}^N$ into the problem (1), where η_i denotes the weight of a data point z_i. The constrained learning problem is reformulated as:

$$\begin{aligned} \theta(\eta) = \arg\min_{\theta} \quad & g_0(\eta \cdot D,\theta) \\ \text{s.t.} \quad & g_j(\eta \cdot D,\theta) \le 0, \quad j = 1,\dots,J, \\ & h_t(\eta \cdot D,\theta) = 0, \quad t = 1,\dots,T. \end{aligned} \tag{3}$$

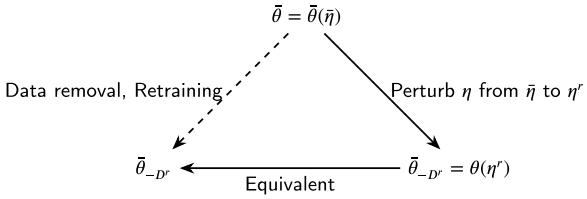


Fig. 1. Illustration of $\bar{\theta}_{-D^r}$ estimation without retraining.

The weight vector $\bar{\eta} = \vec{1}$ recovers the original problem (1), i.e., $\bar{\theta} = \theta(\bar{\eta})$. Removing the data z_i is equivalent to setting its corresponding weight $\eta_i = 0$. We denote the data weight corresponding to the gold standard model (2) as $\eta^r = \{\eta_i^r\}_1^N \in \mathbb{R}^N$, where

$$\eta_i^r = \begin{cases} 0, & \text{if } z_i \in D^r \\ 1, & \text{otherwise} \end{cases}$$

It follows directly that $\bar{\theta}_{-D^r} = \theta(\eta^r)$.

Introducing the weight vector transforms the CMU problem into sensitivity analysis with respect to the weights η. By analyzing how the solution $\theta(\eta)$ changes when the weight vector shifts from the original weight $\bar{\eta} = \vec{1}$ to the weight vector η^r, we can estimate solution $\bar{\theta}_{-D^r} = \theta(\eta^r)$ without retraining. The connection between sensitivity analysis and CMU is depicted in Fig. 1.

3.1. Sensitivity Analysis

This section aims to analyze the sensitivity of the solution θ with respect to the data weights η. We begin by formulating the optimality condition of the data weighted learning problem (3) as a variational inequality (VI). When the data weights η change from $\bar{\eta}$ to η^r, the updated solution $\theta(\eta)$ must jointly satisfy the VI. This fact enables us to estimate the corresponding change in the solution, $\Delta\theta$, given a change in the data weights $\Delta\eta = \eta^r - \bar{\eta}$. We further linearize this VI to reduce computational cost.

We first define the Lagrangian function of data weighted constrained learning (3) as follows:

$$\mathcal{L}(\eta, \theta, \lambda) = g_0(\eta \cdot D, \theta) + \sum_{j=1}^{J} \lambda_g^j g_j(\eta \cdot D, \theta) + \sum_{t=1}^{T} \lambda_h^t h_t(\eta \cdot D, \theta), \tag{4}$$

where $\lambda = [\lambda_g^j, \lambda_h^t]$, $\lambda_g^j \geq 0$ are the Lagrange multipliers associated with the constraints. The VI that captures the associated first-order optimality conditions of (3) is given by (Luo et al., 1996; Facchinei and Pang, 2003):

$$f(\eta, \theta, \lambda) + N_E(\theta, \lambda) \ni 0, \tag{5}$$

where

$$\begin{cases} f(\eta, \theta, \lambda) = \left(\nabla_\theta \mathcal{L}(\eta, \theta, \lambda),\ -\nabla_\lambda \mathcal{L}(\eta, \theta, \lambda)\right)^\top, \\ E = \mathbb{R}^{dim(\theta)} \times \left[\mathbb{R}_+^J \times \mathbb{R}^T\right]. \end{cases} \tag{6}$$

Here $N_E(\theta, \lambda)$ denotes the normal cone to E at (θ, λ); see its definition in Dontchev and Rockafellar (2009) and also listed in Appendix A.2. $\bar{\theta}$ is the optimal solution associated with the full-data weight $\eta = \bar{\eta}$, and we denote its corresponding Lagrange multiplier as $\bar{\lambda}$. $(\bar{\eta}, \bar{\theta}, \bar{\lambda})$ satisfies the VI (5), i.e.,

$$f(\bar{\eta}, \bar{\theta}, \bar{\lambda}) + N_E(\bar{\theta}, \bar{\lambda}) \ni 0. \tag{7}$$

When η shifts from $\bar{\eta}$ to η^r, the change in the solution and Lagrangian multiplier, denoted by $\Delta\theta$ and $\Delta\lambda = [\Delta\lambda_g^j, \Delta\lambda_h^t]$, satisfy the following relation, due to (5):

$$f(\eta^r, \bar{\theta} + \Delta\theta, \bar{\lambda} + \Delta\lambda) + N_E(\bar{\theta} + \Delta\theta, \bar{\lambda} + \Delta\lambda) \ni 0. \tag{8}$$

To reduce computational cost, we linearize the VI (8) as follows:

$$f(\bar{\eta}, \bar{\theta}, \bar{\lambda}) + \nabla_\eta f(\bar{\eta}, \bar{\theta}, \bar{\lambda})(\eta^r - \bar{\eta}) + \nabla_{(\theta,\lambda)} f(\bar{\eta}, \bar{\theta}, \bar{\lambda})[\Delta\theta; \Delta\lambda] + N_E(\bar{\theta} + \Delta\theta, \bar{\lambda} + \Delta\lambda) \ni \mathbf{0}, \tag{9}$$

where $\nabla_{(\theta,\lambda)} f(\bar{\eta}, \bar{\theta}, \bar{\lambda})$ is the Jacobian matrix:

$$\nabla_{(\theta,\lambda)} f(\bar{\eta}, \bar{\theta}, \bar{\lambda}) = \begin{bmatrix} \nabla_\theta^2 \mathcal{L}(\bar{\eta}, \bar{\theta}, \bar{\lambda}), \nabla_{\theta\lambda} \mathcal{L}(\bar{\eta}, \bar{\theta}, \bar{\lambda}) \\ -\nabla_{\theta\lambda} \mathcal{L}(\bar{\eta}, \bar{\theta}, \bar{\lambda}), -\nabla_{\lambda\lambda} \mathcal{L}(\bar{\eta}, \bar{\theta}, \bar{\lambda}) \end{bmatrix}. \tag{10}$$

Linearizing the VI (8) allows us to obtain a first-order (linear) approximation of the change in the solution $\theta(\eta)$ with respect to perturbations in η. As stated in the following theorem, under certain conditions, $\theta(\eta)$ is directionally differentiable and its directional derivative is given by the solution of the linearized VI (9).

Theorem 3.1 (Restatement of Theorem 2E.8 in Dontchev and Rockafellar (2009))**.** *Assume the Linear Independence Constraint Qualification (LICQ) and second-order sufficient conditions (SOSC) hold at* $(\bar{\theta}, \bar{\lambda})$.

Assumption 3.1. (LICQ) The gradients $\nabla_\theta g_j(D,\bar{\theta})$ for the active constraints $(j \in I_{\text{Active}})$ and $\nabla_\theta h_t(D,\bar{\theta})$ are linearly independent.

Assumption 3.2. (SOSC) For any $\Delta\theta \neq 0$ satisfying both $\Delta\theta \perp \nabla_\theta g_j(D,\bar{\theta})$ for all $j \in I_{\text{Active}}$ and $\Delta\theta \perp \nabla_\theta h_t(D,\bar{\theta})$, it holds that $\langle \Delta\theta, \nabla^2_{\theta\theta}\mathcal{L}(\bar{\eta},\bar{\theta},\bar{\lambda})\Delta\theta \rangle > 0$.

Then the directional derivative

$$\lim_{\substack{t\searrow 0^+ \\ \Delta\varepsilon\to\Delta\bar{\eta}}} \frac{\theta(\bar{\eta}+t\Delta\bar{\varepsilon})-\theta(\bar{\eta})}{t}$$

exists and is given by the unique solution of the linearized variational inequality (9).

Remark 3.1. Detailed proof of the theorem can be found in Dontchev and Rockafellar (2009), which is omitted here. Specifically, LICQ means that the gradients of all active inequality constraints together with those of the equality constraints are linearly independent. In the context of PINNs, these constraints correspond to PDE residuals. This condition implies that different PDE constraints should provide non-redundant information. In particular, if multiple PDEs describe similar or overlapping physical laws (e.g., traffic flow conservation), their gradients may become linearly dependent, violating LICQ. Such dependency can lead to ill-conditioned optimization problems and may result in unstable estimation of machine unlearning. The SOSC requires that the Hessian of the Lagrangian is positive definite along all feasible first-order directions that respect the active constraints. We note that in deep neural networks (including PINNs), the loss landscape is inherently nonconvex, and the restricted SOSC may not strictly hold due to saddle points and indefinite curvature. One potential way to deal with this challenge theoretically is to study the generic conditions for optimality in constrained minimization problems (Spingarn, 1978; Spingarn and Rockafellar, 1979). Such generic conditions allow one to define a class of similar problems to the studied problem, and ensure that their second optimality conditions will be satisfied in the same way. This also implies that if one can find a well-behaved problem in the class of problems, the studied problem will also be well-behaved. Of course, how to define the problem class and find a well-behaved problem is the key challenge, which we plan to explore in future research.

To mitigate this difficulty in practice, we adopt a standard damping technique (Koh and Liang, 2017) in our algorithmic implementation. Specifically, we add a damping term to the Hessian matrix (i.e., replacing $\nabla^2_{\theta\theta}L$ with $\nabla^2_{\theta\theta}L + \lambda I$), where $\lambda > 0$ is a tunable hyperparameter. Empirical evidence in Section 5 consistently shows that the proposed approximation remains accurate in local neighborhoods of trained solutions.

Remark 3.2. Under the conditions in Theorem 3.1, the solution mapping with respect to data weights is locally differentiable and enjoys a Lipschitz-like property (Gfrerer and Outrata, 2016; Dontchev and Rockafellar, 2009). This Lipschitz-like property ensures that small perturbations in the data do not lead to abrupt changes in the solution, thereby guaranteeing the smooth dependence of the solution on data weights. As a result, the first-order approximation provided by CMU accurately tracks the retrained solution in a local neighborhood. These conditions are naturally satisfied in many classical machine learning settings. For example, in models with linear kernels and convex loss functions (e.g., SVMs or logistic regression), together with convex constraints, the optimization problem is convex and admits a unique and stable solution. In such cases, the solution mapping with respect to data weights is single-valued, locally differentiable, and Lipschitz continuous, and the first-order approximation provided by CMU is theoretically well-justified. However, these conditions may not hold when small perturbations in the data cause the optimizer to jump to a different local minimum or induce a change in the active constraint set. In such cases, the solution mapping may become non-smooth or even set-valued, violating the assumptions required for first-order sensitivity analysis. An important direction to remedy this is to explicitly treat the solution as a set-valued mapping and study how this solution set evolves under data perturbations or removal (Wang et al., 2025). This perspective may provide a more complete characterization of solution instability in nonconvex or degenerate constrained problems.

Following Theorem 2E.4 in Dontchev and Rockafellar (2009), the VI (9) can be simplified as:

$$\nabla_\eta f(\bar{\eta},\bar{\theta},\bar{\lambda})(\eta^r-\bar{\eta}) + \nabla_{(\theta,\lambda)} f(\bar{\eta},\bar{\theta},\bar{\lambda})[\Delta\theta;\Delta\lambda] + N_{\bar{E}}(\Delta\theta,\Delta\lambda) \ni \mathbf{0}, \tag{11}$$

where

$$\bar{E} = \mathbb{R}^{\dim(\theta)} \times V,$$
$$V := \left\{ \Delta\lambda \in \mathbb{R}^{J+T} \middle| \begin{array}{ll} \Delta\lambda_g^j \geq 0 & \text{for } j \in [1,J] \text{ with } g_j(D,\bar{\theta}) = 0,\ \bar{\lambda}_g^j = 0, \\ \Delta\lambda_g^j = 0 & \text{for } j \in [1,J] \text{ with } g_j(D,\bar{\theta}) < 0. \end{array} \right\}. \tag{12}$$

Here, V denotes the admissible set of perturbations of the Lagrange multipliers. In particular, V enforces that multiplier perturbations corresponding to strictly inactive inequality constraints remain zero, while those associated with active constraints with zero multipliers are restricted to nonnegative directions.

By solving the VI (11), we obtain the changes in the solution and Lagrange multipliers, i.e., $\Delta\theta$ and $\Delta\lambda$. The updated solution after data removal can then be approximated by $\theta_{-D^r} \approx \bar{\theta} + \Delta\theta$. In Section 4.3 below, we reformulate the VI (11) as a quadratic program to facilitate numerical optimization in the context of learning-based ITS.

4. Constrained Machine Unlearning for Learning-based ITS Models

Data-driven ITS based on ML and DL admits a wide variety of formulations depending on specific application scenarios. Nevertheless, most learning-based ITS models can ultimately be cast as the minimization of an empirical risk objective, possibly subject

Table 2
Examples of constraints in ML-based ITS models.

Constraint Source	Application	Constraint Formulation	Aim of Constraint
ML model-driven	Classify vehicle type using SVM (Sun and Ban, 2013)	Geometric margin of at least 1 for each training point	Ensure a minimum margin for better generalization
	Traffic density field reconstruction using partially observed data (Huang and Agarwal, 2022; Shi et al., 2021)	Residual value of the PDE in traffic flow model equals to zero	Enforce that the estimated density respects the Greenshields-based LWR
Domain Knowledge or Physical Models	Estimate high-resolution speed fields from sparse probe data (Thodi et al., 2022)	Limit the model to use information only from directions in which traffic waves realistically propagate	Characterize the wave propagation
	Vehicle longitudinal trajectory prediction on highways (Geng et al., 2023)	The predicted trajectory coordinates must follow the Intelligent Driver Model	Model the Car Following (CF) behavior of the subject vehicle relative to its leader
Stability-motivated	Network-level speed forecasting (Cui et al., 2019)	L1 norm/L2 norm constraints on the weights of graph neural network	Reduce model complexity, prevent overfitting
Safety-motivated	Path prediction for multiple interacting agents (Sadeghian et al., 2019)	Embedded physical information extracted from scene images	Ensure physical feasibility and safety
	Traffic signal control (Barhoumi et al., 2025)	Requiring the time to collision between vehicles to exceed a predefined safety threshold.	Ensure collision-free vehicle interactions during learning
Fairness-motivated	Traffic speed estimation under imbalanced sensor distribution (Xia et al., 2025)	Disparities in predictive performance between spatial regions	Reduce variance in predictions across areas with uneven sensor coverage

to operational or physical constraints. Therefore, we adopt this unified empirical risk minimization framework as the foundation for our CMU model, to cover a broad range of data-driven ITS applications.

This section instantiates the objective function and constraints of the general constrained learning problem (1) in the context of ITS, and presents a numerical approach for the CMU method introduced in Section 3.

4.1. Learning-based ITS Models

In practical applications, especially with large-scale or high-dimensional data, ITS models are learned directly from data using empirical loss minimization approaches. The objective function is typically formulated as the sum of individual data losses, i.e., $\sum_1^N \ell(z_i, \theta)$. For example, $\ell(z_i, \theta) = \|f_\theta(x_i, t_i) - z_i\|_2$ measures the discrepancy between the estimated traffic state produced by an ML model f_θ and the ground truth observation z_i at location x_i and time t_i.

The types of constraints in ITS can vary widely. They may encode domain-specific knowledge (e.g., traffic flow theory) or serve to enforce desirable properties such as fairness or robustness. For example, Xia et al. (2025) introduced region-based static fairness to address the prediction performance gaps between regions caused by uneven sensor deployment. Huang and Agarwal (2022) introduced the LWR residual error as a physical loss to balance predictive accuracy with theoretical consistency. We briefly summarize the applications and purposes of these constraints in Table 2. Note that ITS models often contain "nominal" constraints, such as flow conservation or nonnegativity, which are omitted from Table 2 for brevity. Specifically, some research works adopt the regularization techniques to embed knowledge or properties. We treat them as penalty-based relaxations of constrained optimization.

To encompass the diverse ITS cases in Table 2, we customize the constrained learning problem (1) as the ITS model (13). The objective function $g_0(D, \theta)$ is expressed as the sum of individual loss terms over the dataset. Since each data point z_i may contribute to one or more constraints, and not every constraint depends on all data points, we define the inequality constraints as $g_j\left(z_{I_j}, \theta\right) \le 0$ and equality constraints as $h_t\left(z_{I_t}, \theta\right) = 0$. Here, $z_{I_j} \subset D$ and $z_{I_t} \subset D$ denote subsets of data points associated with the j-th inequality and t-th equality constraint, respectively.

$$\begin{aligned} \bar{\theta} = \arg\min_{\theta} & \sum_{i=1}^{N} \ell\left(z_i, \theta\right) \\ \text{s.t.} \quad & g_j\left(z_{I_j}, \theta\right) \le 0, \quad \forall j \in \{1, \dots, J\}, \\ & h_t\left(z_{I_t}, \theta\right) = 0, \quad \forall t \in \{1, \dots, T\}. \end{aligned} \tag{13}$$

J and T denote the total numbers of inequality and equality constraints, respectively. We focus on removing one data point upon receiving the data deletion request. Without loss of generality, we assume that the deletion request involves only the last data point z_N, and z_N contributes exclusively to the J-th inequality constraint and the T-th equality constraint. A general case where multiple data points are removed is provided in Appendix C.

After removing z_N, the data-driven ITS problem (13) reduces to the following gold standard model:

$$\begin{aligned}\bar{\theta}_{-z_N} = \arg\min_{\theta} \sum_{i=1}^{N-1} \ell\left(z_i, \theta\right) \\ \text{s.t.} \quad g_j\left(z_{I_j}, \theta\right) \le 0, \quad \forall j \in \{1, \dots, J-1\}, \\ g_J\left(z_{I_J} \setminus \{z_N\}, \theta\right) \le 0, \\ h_t\left(z_{I_t}, \theta\right) = 0, \quad \forall t \in \{1, \dots, T-1\}, \\ h_T\left(z_{I_T} \setminus \{z_N\}, \theta\right) = 0.\end{aligned} \tag{14}$$

If a constraint only involves z_N, it will not be enforced. The challenge is to approximate the updated solution $\bar{\theta}_{-z_N}$ based on $\bar{\theta}$ without retraining while ensuring it satisfies the constraints associated with the remaining data points.

4.2. Data Weighted ITS Model

To efficiently address the influence of z_N, we propose incorporating the constraints involved with z_N directly into the objective function using a *penalty* function. We call $\phi(C,t)$ a penalty function if it is continuous, non-negative, satisfies $\phi(C,t) = 0$ for $t \le 0$, and is strictly increasing with respect to both $C > 0$ and $t > 0$. For example, a penalty function would be $\phi(C,t) = \begin{cases} 0 & \text{for } t < 0 \\ Ct^3 & \text{for } t \ge 0 \end{cases}$.

The original ITS (13) is reformulated as:

$$\begin{aligned}\bar{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} \ell(z_i, \theta) + \phi\left(C_g, g_J(z_{I_J}, \theta)\right) + \phi\left(C_h, h_T(z_{I_T}, \theta)\right) + \phi\left(C_h, -h_T(z_{I_T}, \theta)\right) \\ \text{s.t.} \quad g_j(z_{I_j}, \theta) \le 0, \quad \forall j \in \{1, \dots, J-1\}, \\ h_t(z_{I_t}, \theta) = 0, \quad \forall t \in \{1, \dots, T-1\}.\end{aligned} \tag{15}$$

Where C_h, C_g are the penalty constants that penalize violations of the z_N-related constraints.

Following the data weighted constrained learning (3), we introduce a weight η_N for the terms associated with z_N, allowing us to remove the influence of z_N by decreasing the weight η_N gradually. The *data weighted ITS model* is formulated as follows:

$$\begin{aligned}\theta(\eta_N) = \arg\min_{\theta} \Bigg\{ \sum_{i=1}^{N-1} \ell(z_i, \theta) + \eta_N \cdot \ell(z_N, \theta) + \phi\left(C_g, g_J(z_{I_J} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) \\ + \phi\left(C_h, h_T(z_{I_T} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) + \phi\left(C_h, -h_T(z_{I_T} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) \Bigg\} \\ \text{s.t.} \quad g_j(z_{I_j}, \theta) \le 0, \quad \forall j \in \{1, \dots, J-1\}, \\ h_t(z_{I_t}, \theta) = 0, \quad \forall t \in \{1, \dots, T-1\}.\end{aligned} \tag{16}$$

Here η_N acts as a tunable parameter that controls the contribution of the terms associated with z_N in the optimization problem. When $\eta_N = 1$, the data weighted ITS (16) reduces to the original ITS problem (13), as z_N is fully included with its original weight. When $\eta_N = 0$, the terms related to z_N are completely removed, and the problem (16) is equivalent to the gold standard model (14).

4.3. Auxiliary Problem

Denoting the Lagrangian function of data weighted ITS model (16) as $L(\eta_N, \theta, \lambda)$, we define $D = \left[z_1, z_2, \dots, z_N\right]$ to represent the data points and $\lambda = \left[\lambda_g, \lambda_h\right]$ as the vector of Lagrange multipliers. Here, $\lambda_g = \left[\lambda_g^j\right]$ corresponds to the multipliers associated with the inequality constraints $g_j(z_{I_j}, \theta) \le 0$, and $\lambda_h = \left[\lambda_h^t\right]$ corresponds to the multipliers associated with the equality constraints $h_t(z_{I_t}, \theta)$. The Lagrangian function is given by:

$$\begin{aligned}L(\eta_N, \theta, \lambda) = \sum_{i=1}^{N-1} \ell(z_i, \theta) + \eta_N \cdot \ell(z_N, \theta) + \phi\left(C_g, g_J(z_{I_J} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) \\ + \phi\left(C_h, h_T(z_{I_T} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) + \phi\left(C_h, -h_T(z_{I_T} \setminus \{z_N\}, \eta_N \cdot z_N, \theta)\right) \\ + \sum_{j=1}^{J-1} \lambda_g^j g_j\left(z_{I_j}, \theta\right) + \sum_{t=1}^{T-1} \lambda_h^t h_t\left(z_{I_t}, \theta\right).\end{aligned} \tag{17}$$

Given problem (13) with its optimal solution $\bar{\theta}$ and the corresponding Lagrangian multiplier $\bar{\lambda}$, we have $\left[\bar{\theta}, \bar{\lambda}\right] = \left[\theta(\eta_N = 1), \lambda(\eta_N = 1)\right]$. Following Theorem 3.1 in Section 3.1, conducting the sensitivity of θ w.r.t. η_N ultimately reduces to solving the following linearized

VI:

$$\nabla_{\eta_N} f(\bar{\eta}_N, \bar{\theta}, \bar{\lambda})\Delta\eta_N + \nabla_{(\theta,\lambda)} f(\bar{\eta}_N, \bar{\theta}, \bar{\lambda})[\Delta\theta; \Delta\lambda] + N_{\bar{E}}(\Delta\theta, \Delta\lambda) \ni \mathbf{0}, \tag{18}$$

where

$$f(\eta_N, \theta, \lambda) = \big(\nabla_\theta L(\eta_N, \theta, \lambda), -\nabla_\lambda L(\eta_N, \theta, \lambda)\big)^\top \tag{19}$$

and

$$\bar{E} = \mathbb{R}^{\dim(\theta)} \times V,$$
$$V := \left\{ \Delta\lambda \in \mathbb{R}^{J+T-2} \,\middle|\, \begin{array}{ll} \Delta\lambda_g^j \geq 0 & \text{for } j \in [1, J-1] \text{ with } g_j(I_j, \bar{\theta}) = 0, \bar{\lambda}_g^j = 0, \\ \Delta\lambda_g^j = 0 & \text{for } j \in [1, J-1] \text{ with } g_j(I_i, \bar{\theta}) < 0. \end{array} \right\}. \tag{20}$$

The solution $(\Delta\theta, \Delta\lambda)$ of the VI (18) approximates the changes in the primal and dual variables of the ITS model (13) induced by the removal of z_N.

However, the VI formulation is not well-suited for a direct numerical algorithm and thus requires further reformulation. Next, we transfer the VI (18) to a quadratic program, referred to as the *Auxiliary Problem*, to solve for $\Delta\theta$. The auxiliary problem (21) is designed to measure $\Delta\theta$, as η_N moves away from $\bar{\eta}_N = 1$ to 0. We denote the move direction as $q = -1$.

$$\begin{aligned} \min_{\Delta\theta} \quad & \bar{g}_0(\Delta\theta) + \langle \nabla^2_{\theta\eta_N} L(\bar{\eta}_N, \bar{\theta}, \bar{\lambda}) q, \Delta\theta \rangle \\ \text{s.t.} \quad & \bar{g}_j(\Delta\theta, z_{I_j}) \begin{cases} = 0 & \text{for } i \in I \setminus I_0, \\ \leq 0 & \text{for } i \in I_0, \end{cases} \\ & \bar{h}_t(\Delta\theta, z_{I_t}) = 0, \quad t \in \{1, 2, \ldots, T-1\}. \end{aligned} \tag{21}$$

where the index sets I and I_0 are defined as:

$$I = \Big\{ j \in [1, J-1] \mid g_j(z_{I_j}, \bar{\theta}) = 0 \Big\},$$
$$I_0 = \Big\{ j \in [1, J-1] \mid g_j(z_{I_j}, \bar{\theta}) = 0 \text{ and } \bar{\lambda}_g^j = 0 \Big\} \subset I.$$

Additionally, $\bar{g}_0(\Delta\theta)$ is the second order expansion of $L(\eta_N, \theta, \lambda)$:

$$\bar{g}_0(\Delta\theta) = L(\bar{\eta}_N, \bar{\theta}, \bar{\lambda}) + \big\langle \nabla_\theta L(\bar{\eta}_N, \bar{\theta}, \bar{\lambda}), \Delta\theta \big\rangle + \frac{1}{2} \big\langle \Delta\theta, \nabla^2_{\theta\theta} L(\bar{\eta}_N, \bar{\theta}, \bar{\lambda})\Delta\theta \big\rangle, \tag{22}$$

and $\bar{g}_j(\Delta\theta, z_{I_j}), \bar{h}_t(\Delta\theta, z_{I_t})$ are the first-order expansion of constraints:

$$\bar{g}_j(\Delta\theta, z_{I_j}) = g_j(z_{I_j}, \bar{\theta}) + \Big\langle \nabla_\theta g_j(z_{I_j}, \bar{\theta}), \Delta\theta \Big\rangle, \bar{h}_t(\Delta\theta, z_{I_t}) = h_t(z_{I_t}, \bar{\theta}) + \Big\langle \nabla_\theta h_t(z_{I_t}, \bar{\theta}), \Delta\theta \Big\rangle. \tag{23}$$

Theorem 4.1. *The auxiliary problem (21) admits the same solution as VI (18). In particular, the minimizer $\Delta\theta$ of the auxiliary problem (21) also satisfies the VI (18).*

Proof. See Appendix A.3, where we show that the first-order optimality conditions of the auxiliary problem coincide with the variational inequality (18). □

By solving the auxiliary problem (21) for $\Delta\theta$, we can determine how θ changes from $\bar{\theta}$ when η_N moves from $\bar{\eta}_N$ along the direction of q. Also note that (22) is a quadratic function and (23) is a linear function. Therefore, problem (21) is a quadratic optimization problem with linear constraints, which is normally much easier to solve than retraining the original ITS model.

4.4. Special Case: Unconstrained Setting

When the ITS model has no constraints, our method can be simplified. For example, assume the data weighted ITS model (16) is

$$\bar{\theta} = \arg\min_\theta \sum_{i=1}^{N-1} \ell\big(z_i, \theta\big) + \eta_N \cdot \nabla_\theta \ell(z_N, \theta). \tag{24}$$

The optimality condition reduces to

$$\sum_{i=1}^{N-1} \nabla_\theta \ell\big(z_i, \bar{\theta}\big) + \eta_N \cdot \ell(z_N, \bar{\theta}) = \mathbf{0}. \tag{25}$$

The influence of data point z_N on the solution $\bar{\theta}$ can be measured by its derivative, which is derived using the Dini implicit function theorem (Krantz and Parks, 2002):

$$\frac{\partial\theta}{\partial\eta_N}\bigg|_{\eta_N=1} = -\left[\sum_{i=1}^{N} \nabla^2_\theta \ell\big(z_i, \bar{\theta}\big)\right]^{-1} \nabla_\theta \ell\big(z_N, \bar{\theta}\big). \tag{26}$$

The $\Delta\theta$ can be estimated by:

$$\Delta\theta = \theta(\eta_N = 0) - \theta(\eta_N = 1) \approx \frac{\partial\theta}{\partial\eta_N}|_{\eta_N=1} \cdot (-1). \tag{27}$$

When constraints are introduced, the optimality condition can be characterized by the Karush-Kuhn-Tucker (KKT) conditions, rendering the Dini implicit function theorem inapplicable. Moreover, the Dini implicit function theorem requires the loss function to have a non-singular Hessian matrix, a condition that is often violated in deep learning scenarios due to the non-convex nature of neural networks. By leveraging the generalized implicit function theorem proposed by Dontchev and Rockafellar (2009), our proposed CMU methods can address these issues and thus extend the influence function to constrained learning problems. This extension further enables the simultaneous removal of multiple data points; see Appendix C.

The overall workflow of the proposed algorithm is summarized in Algorithm 1.

Algorithm 1 CMU for ITS Models.

Require: Trained ITS model with parameters $\bar{\theta}$; training dataset $D = \{z_1, \dots, z_N\}$; data removal request z_N; constraint functions $\{g_j, h_t\}$; penalty parameters (C_g, C_h).

Ensure: Unlearned model parameters $\bar{\theta}_{-z_N}$.

1: **Data-weighted modeling:** Introduce data weights η_N for z_N, and reformulate the ITS model as the data-weighted problem (16).

2: **Reference point:** Set $\bar{\eta}_N = \mathbf{1}$ and treat $(\bar{\theta}, \bar{\lambda}) = (\theta(\bar{\eta}_N), \lambda(\bar{\eta}_N))$ as the reference solution.

3: **Sensitivity direction:** Define the removal direction $q = -\mathbf{1}$, corresponding to decreasing η_N from 1 to 0 and fully removing the data point z_N.

4: **Auxiliary problem construction:** Construct the auxiliary quadratic program (21).

5: **Solve auxiliary problem:** Compute the parameter update $\Delta\theta$ by solving the quadratic program.

6: **Model update:** Obtain the unlearned model parameters via

$$\theta_{-z_N} \approx \bar{\theta} + \Delta\theta.$$

7: **return** θ_{-z_N}.

5. Application

We implement our CMU model and algorithm on two ITS applications using trajectory data. The first one is a support vector machine (SVM) based vehicle classification model that utilizes GPS trajectories to distinguish between passenger cars and trucks. The second application employs a physics-informed neural network (PINN) to reconstruct vehicle velocity fields using the Next Generation Simulation (NGSIM) dataset.

Given that trajectory data often contains sensitive personal information–such as home and work locations, travel habits, and frequently visited places–many jurisdictions mandate that organizations allow users to opt out of data collection or request data erasure. Additionally, malicious actors can inject poisoned trajectory data to manipulate model predictions, potentially fabricating congestion patterns or altering autonomous vehicle behavior. When any of these scenarios occur, the sensitive or poisoned data points may need to be removed, leading to the machine unlearning problem as discussed here. In this case, our approach ensures compliance with privacy regulations and enhances model integrity against adversarial attacks in a computationally efficient manner.

5.1. Constrained Machine Unlearning of SVM

We evaluate our method on an SVM-based vehicle classification model. The classification task utilizes acceleration and deceleration features derived from real-world vehicle GPS trajectories to divide vehicles into passenger cars or trucks. Details about the model and dataset can be found in (Sun and Ban, 2013).

5.1.1. Data Weighted SVM

Denoting the data as $z_i = [x_i, y_i]$, where $i \in \{1, 2, \dots, N\}$, x_i represents the feature vector, and $y_i \in \{-1, 1\}$ is the label, the soft-margin SVM model can be formulated as:

$$\begin{aligned} \min_{\theta=\{w,b,\xi\}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\eta \cdot x_i + b) \geqslant 1 - \xi_i, \\ & \xi_i \geqslant 0. \\ & i = 1, \dots, N \end{aligned} \tag{28}$$

where $C > 0$ is the regularization parameter penalizing the slack variables ξ_i.

Table 3
Comparison of SVM Models After Unlearning.

Model	w_1	w_2	b	Train Accuracy	Test Accuracy
Original Model	-7.4230	-2.6563	5.0319	0.9667	0.9333
CMU	-7.3818	-1.8735	4.5935	0.9667	0.9500
Gold Standard Model	-7.3402	-1.6735	4.4707	0.9661	0.9500

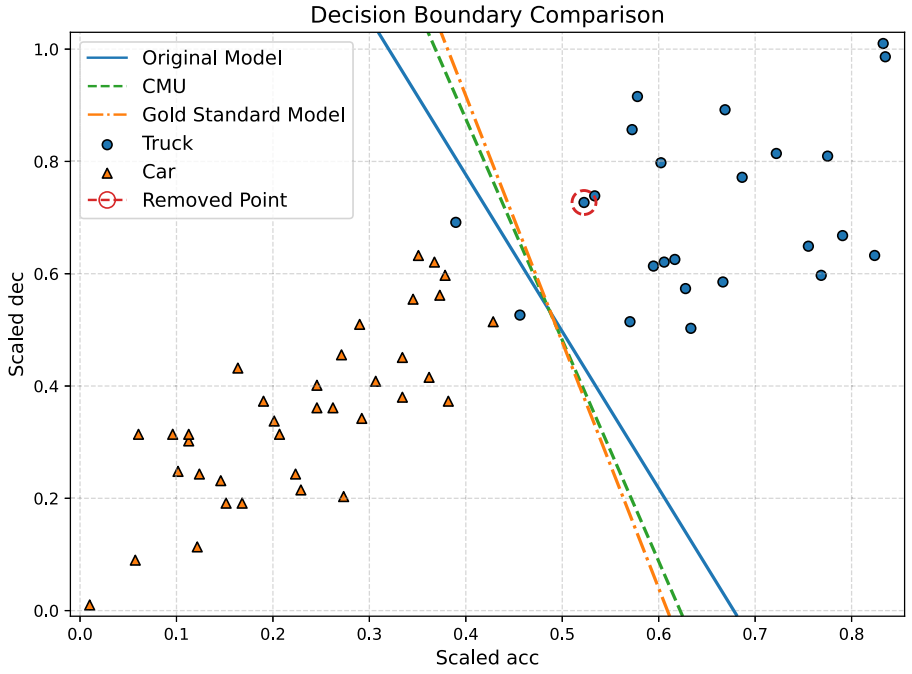


Fig. 2. Comparison of Decision Boundaries: Original SVM, CMU, and Gold Standard.

In this section, we assume we aim to remove the last data z_N. Following the data weighted ITS model (16), the data weighted SVM model is:

$$\begin{aligned} \min_{\theta=\{w,b,\xi\}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N-1}\xi_i + \eta_N \cdot C\max\left[0, 1 - y_N\left(w^\top x_N + b\right)\right] \\ \text{s.t} \quad & y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \\ & i = 1, \ldots, N-1. \end{aligned} \tag{29}$$

It is straightforward to verify that model (Data Weighted SVM) is equivalent to model (28) when $\eta_N = 1$, and is equivalent to the SVM model trained on the remaining $N-1$ data points (gold standard model) when $\eta_N = 0$. By estimating how the solution changes as we decrease η_N from 1 to 0, we can fulfill the data removal request. The sensitivity analysis of solution $\theta = \{w, b, \xi\}$ w.r.t. η_N can be found in the appendix B.

5.2. Results

We first retrain the model after data removal to obtain the gold standard model (ground truth). Subsequently, we calculate the solution change using the constrained machine unlearning approach described in Section 4. Table 3 summarizes the parameters and classification accuracy of the original model, CMU, and the gold standard model. Fig. 2 illustrates the decision boundaries of the different models. First, the SVM solution undergoes significant changes due to the removal of just one data point. This underscores the importance of developing efficient CMU algorithms to handle the data removal request. Second, both the plotted decision boundaries and the numerical comparisons underscore the accuracy of the proposed CMU method in closely approximating the gold standard solution. These results validate the ability of the proposed CMU model to maintain model accuracy while effectively removing the influence of specific data points. Further experiments on the removal of multiple data points are provided in Appendix D. Appendix E proposes a weight-decay method that may potentially deal with deleting larger size data.

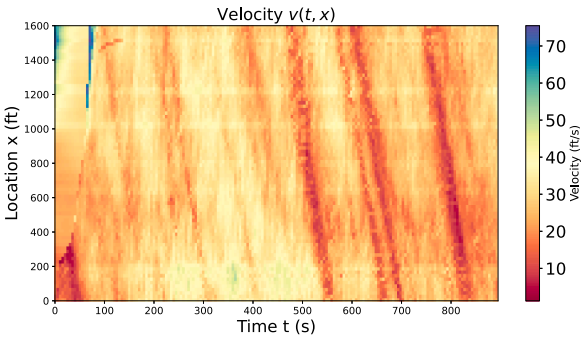


Fig. 3. Visualization of the vehicle velocity field on I-80 from NGSIM.

5.3. Constrained Machine Unlearning for PINN-Based Traffic State Estimation

5.3.1. PINN-Based Traffic State Estimation

The PINN model we test here is based on Shi et al. (2021), which reconstructs the velocity field using the observed vehicle velocity data extracted from the NGSIM dataset. By integrating physical constraints with data-driven learning, the model ensures that the reconstructed velocity field adheres to the LWR model.

Data Description: We extract vehicle trajectory data from the NGSIM dataset for a road segment on I-80 in Emeryville, California, which spans approximately $L = 1600$ feet. The set of vehicle trajectories is represented by $\mathcal{T} = \{(x_i, t_i, v_i)\}$, where each entry represents a vehicle's position x_i, timestamp t_i, and speed v_i. The trajectory to be removed is $\mathcal{T}^r$. The total duration of the data is $T = 15$ minutes.

Data Processing: Following the data processing in Huang and Agarwal (2022), we construct the velocity field using a binning method with a spatial resolution Δx of 20 feet and a temporal resolution Δt of 5 seconds. The center of each cell is treated as a grid point. We denote the spatial-temporal grid by $\Omega = \{(x_p, t_p) | p = 1, 2, \dots, N_p\}$, where x_p and t_p represent the spatial and temporal positions of the grid points, and $N_p = \frac{L}{\Delta x} \times \frac{T}{\Delta t}$ is the total number of grid points. The average speed of vehicles is computed for each bin, resulting in a velocity field consisting of 180 temporal bins and 80 spatial bins. The mean speed in each bin is computed as:

$$v(x_p, t_p) = \frac{1}{\left|S(x_p, t_p)\right|} \sum_{(x_i, t_i, v_i) \in S(x_p, t_p)} v_i \tag{30}$$

where $S(x_p, t_p) = \left\{ (x_i, t_i, v_i) \in \mathcal{T} \mid x_i \in \left[x_p - \frac{1}{2}\Delta x, x_p + \frac{1}{2}\Delta x\right], t_i \in \left[t_p - \frac{1}{2}\Delta t, t_p + \frac{1}{2}\Delta t\right] \right\}$ is the set of trajectory data points that fall within the spatial-temporal bin. $\left|S(x_p, t_p)\right|$ is the number of data points in the bin. Only a subset of the velocity field (14% of samples) will be used as *observed data* to train the model, while the remaining values will serve as the ground truth. These unobserved values are held out from training and used as test data during evaluation. We denote the observed data index as $\mathbf{O} = \{(x_o, t_o) | o = 1, 2, \dots, N_o\}$. The observed data $v(x, t), (x, t) \in \mathbf{O}$ is used to reconstruct the whole velocity field. We also choose a subset of the grid points $\mathbf{A} = \{(x_a, t_a) | a = 1, 2, \dots, N_a\}$, as the *auxiliary points* to examine the physical constraints. Fig. 3 shows the velocity field on the I-80 freeway from 4:00 p.m. to 4:15 p.m., based on the NGSIM dataset. The presence of shock waves can also be observed.

PINN Model Details: We follow the PINN model proposed by Shi et al. (2021). The goal of PINN is to learn a neural network $\hat{v}(x, t; \theta)$, where (x, t) are the inputs and θ represents the trainable parameters, to approximate $v(x, t)$. At the same time, $\hat{v}(x, t; \theta)$, as the traffic state estimator, must adhere to the fundamental dynamics of traffic flow. These dynamics are governed by the LWR model. We assume that density ρ, flow q, and speed v follow Greenshields' fundamental diagram. To facilitate our discussion, the PDE of the LWR is given by:

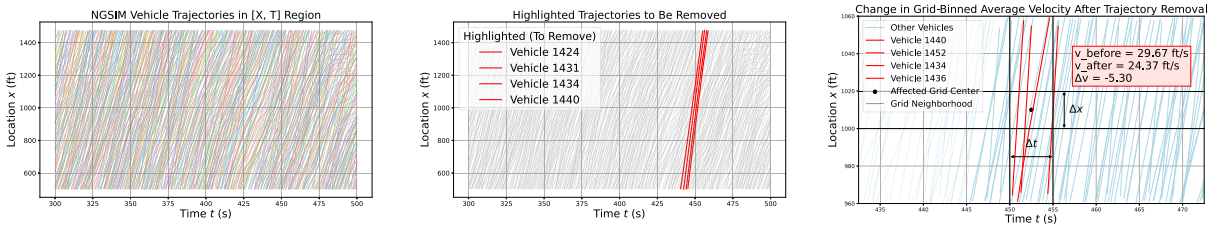
$$\mathcal{N}(\hat{v}(x, t, \theta), Q(\hat{v}(x, t, \theta)); \omega) = 0, x \in [0, L], t \in [0, T], \tag{31}$$

where $\omega := \{v_f, \rho_m\}$ denotes the given free-flow speed and jam density, and $Q(\hat{v})$ denotes the intermediate unobserved traffic variables (flow, density) that have some hidden relationship with $\hat{v}$.

5.3.2. Data Weighted PINN Model

The **original PINN** can be formulated as:

$$\begin{aligned} \min_{\theta} \quad & \sum_{(x_o, t_o) \in \mathbf{O}} \left\| \hat{v}(x_o, t_o, \theta) - v(x_o, t_o) \right\|_2^2, \\ \text{s.t.} \quad & \mathcal{N}(\hat{v}(x_a, t_a, \theta), Q(\hat{v}(x_a, t_a, \theta)); \omega) = 0, \qquad \forall (x_a, t_a) \in \mathbf{A} \\ & v(x_o, t_o) = \frac{1}{\left|S(x_o, t_o)\right|} \sum_{(x_i, t_i, v_i) \in S(x_o, t_o)} v_i. \end{aligned} \tag{32}$$



(a) (b) (c)

Fig. 4. Illustration of trajectory removal and its impact on observed velocity. (a) All trajectories. (b) Subset of removed trajectories. (c) Change in average velocity in affected spatiotemporal bins due to trajectory removal.

Table 4
Performance Comparison of Different PINN Models on the Residual Dataset After Removal.

Model	MAE (Data Loss)	MAE (Physics Loss)	Relative L_2 Error (%)	Training Time (s)
Original Model	1.43	2.1×10^{-2}	8.46	327.06
Retrained Model	2.495	1.12×10^{-2}	14.24	331.50
CMU	2.832	0.37×10^{-2}	16.68	91.77

We use $\mathcal{T}^r \subset \mathcal{T}$ to denote the trajectory data requested to be removed and define

$$S^r(x_p,t_p) = \left\{ (x_i,t_i,v_i) \in \mathcal{T}^r \mid x_i \in \left[x_p - \frac{1}{2}\Delta x, x_p + \frac{1}{2}\Delta x\right], t_i \in \left[t_p - \frac{1}{2}\Delta t, t_p + \frac{1}{2}\Delta t\right] \right\}$$

as the set of removed trajectory data that fall within the spatial-temporal cell around (x_p,t_p). The mean speed at (x_p,t_p), defined by (30) is reformulated as:

$$v(x_p,t_p,\eta) = \frac{1}{\left|S^k(x_p,t_p)\right| + \eta\left|S^r(x_p,t_p)\right|}\left(\sum_{(x_i,t_i,v_i)\in S^k(x_p,t_p)} v_i + \eta \sum_{(x_i,t_i,v_i)\in S^r(x_p,t_p)} v_i\right), \tag{33}$$

where $S^k(x_p,t_p) = S(x_p,t_p) \setminus S^r(x_p,t_p)$. η denotes the weight assigned to the trajectory data that needs to be removed. The **Data-weighted PINN model** is formulated as:

$$\begin{aligned} \min_{\theta} \quad & \sum_{(x_o,t_o)\in\mathbf{O}} \left\| M(x_o,t_o,\theta) - v_w(x_o,t_o) \right\|_2^2, \\ \text{s.t.} \quad & \mathcal{N}(M(x_a,t_a,\theta), Q(M(x_a,t_a,\theta)); \omega) = 0, \forall (x_a,t_a) \in \mathbf{A} \\ & v_w(x_o,t_o) = \frac{1}{\left|S^k(x_o,t_o)\right| + \eta\left|S^r(x_o,t_o)\right|}\left(\sum_{(x_i,t_i,v_i)\in S^k(x_o,t_o)} v_i + \eta \sum_{(x_i,t_i,v_i)\in S^r(x_o,t_o)} v_i\right). \end{aligned} \tag{34}$$

It is trivial to see the problem (34) is equivalent to the original PINN (32) when $\eta = 1$. When η shifts from 1 to 0, the problem (34) shifts to the PINN trained on the remaining data $\mathcal{T} \setminus \mathcal{T}^r$.

5.3.3. Experiment on PINN

Experimental setting: We consider removing 10% of the training data, which results in modifications to the observed average speed in certain spatiotemporal bins. Our CMU method adjusts the original PINN parameters to accommodate this change, and the updated model is then compared with the gold standard model by retraining. Fig. 4 details the data removal workflow. All experiments are conducted on an RTX 4090 GPU.

Table 4 presents the performance of three PINN models–Original, Retrained, and CMU–on the residual dataset after data removal. The Retrained model achieves the best performance across all metrics, with the lowest data loss (MAE: 2.495), physics loss (MAE: 1.12×10^{-2}), and relative L_2 error (14.24%). CMU shows slightly higher errors than the Retrained Model, but remains significantly better than the Original Model. Notably, CMU requires only 91.77 seconds to update the parameters, which is approximately 3.6× faster than full retraining, demonstrating the computational advantage of our CMU approach. Note that the PINN used in our experiments is relatively small, consisting of nine layers with approximately 10,000 parameters. Even for such a compact DNN, CMU can save up to three times the training time. For larger neural networks, where the cost of full retraining is significantly higher, the CMU method is expected to offer even greater time savings.

Fig. 5 illustrates the predicted velocity fields and their differences after removing 10% of the trajectories. The left column shows the velocity fields predicted by the CMU, Original, and Retrained models, respectively. The right column visualizes the absolute differences in predicted speeds: the top figure compares the CMU and Original models, while the bottom figure compares the Retrained (gold

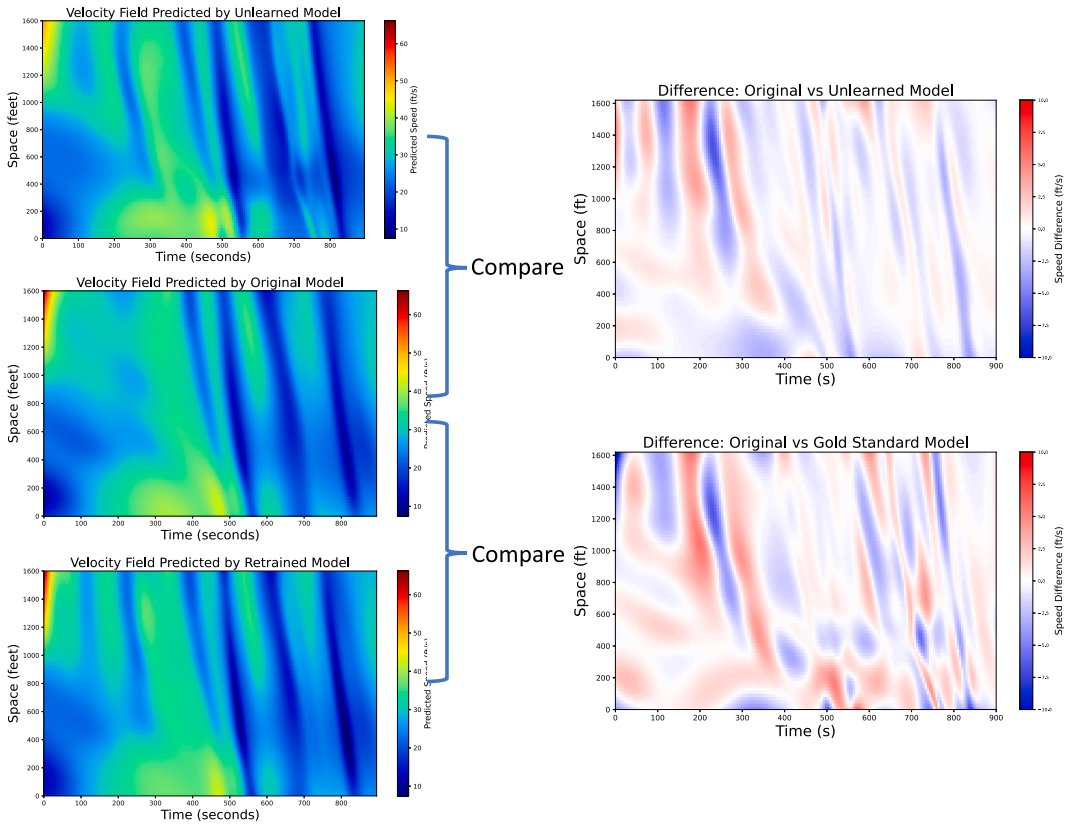


Fig. 5. Predicted velocity field differences after model unlearning and retraining (10% data removed).

standard) and Original models. The difference maps reveal that the CMU model achieves a comparable correction effect to that of the fully Retrained model, but with significantly lower computational cost. These results confirm the effectiveness of the CMU approach in adapting the model to data removal while maintaining high fidelity in the reconstructed velocity field.

6. Conclusions and Future Research

This paper proposes a constrained machine unlearning (CMU) framework tailored to ITS applications. The goal is to enable ITS models to forget privacy-sensitive, poisoned, or outdated data without retraining from scratch. Building upon the influence function concept, we develop a sensitivity analysis framework to approximate the change in optimal model solution resulting from the removal of one or more data points. To the best of our knowledge, this is the first machine unlearning algorithm specifically designed for constrained learning problems in ITS.

The proposed method is further instantiated in two representative learning paradigms: (i) support vector machines (SVMs), where inequality constraints encode margin conditions, and (ii) physics-informed neural networks (PINNs), where the learned solution must respect the traffic flow theory. In both cases, the proposed CMU method significantly reduces computational cost compared to the gold standard model, while preserving key performance metrics. Overall, this work bridges the gap between machine unlearning on ITS, robust statistics, and sensitivity analysis, offering a novel perspective on privacy-preserving, safety-aware, and computationally efficient learning pipelines.

Future research should focus on improving both the theoretical rigor and practical validation of the proposed CMU framework. As CMU is based on local analysis techniques, it would be helpful to establish a more theoretically rigorous bound on the deletion size of the training dataset beyond which CMU may not work well or establish the convergence conditions for the weight-decay method in Appendix E. In addition, the generic conditions (Spingarn, 1978) for the first-order and second-order conditions in Theorem 3.1 can be explored, which may provide new insights to check these conditions for complex DL models. Furthermore, when the conditions in Theorem 3.1 do not hold, a potential way to remedy this is to treat the solution mapping (from data weights to model solution) as set-valued and conduct set-valued analysis on how data removal changes the solution set. We have conducted preliminary investigations on this (Wang et al., 2025), which needs to be extended to more general ML/DL settings. CMU also needs to be tested and validated in larger and more complex DL structures to evaluate its performance in both solution quality and computational efficiency.

While we focus on offline CMU scenarios in this paper, future work may extend the proposed methods to handle streaming traffic data, where traffic models must adapt dynamically to forget obsolete or withdrawn data. For this, prior solutions and intermediate

optimization states may be reused to facilitate efficient unlearning. The last direction is to explore the use of CMU to defend against adversarial attacks on ITS applications. For example, carefully crafted adversarial perturbations can be injected into the training dataset of traffic prediction models (i.e., poisoning attacks), which may lead to significant degradation in model performance or even safety critical failures. Existing defense methods often rely on adversarial training or robust optimization, which may be computationally expensive and hard to generalize. CMU offers an alternative line of defense: once adversarial or suspicious data is detected, the model can "forget" the influence of these harmful inputs without full retraining. This reactive approach may complement existing defenses and be particularly useful in scenarios where attacks are detected post hoc or evolve over time.

CRediT authorship contribution statement

Xin Wang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation; **R. Tyrrell Rockafellar:** Writing – review & editing, Methodology, Formal analysis; **Xuegang (Jeff) Ban:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

None.

Acknowledgement

This research is partially supported by the US National Science Foundation (NSF) grants CNS-2034615 and CMMI-2326340. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF.

Appendix A. Mathematical details

A.1. Notation Summary

The notation summary is provided in Table A.1.

Table A.1
Summary of Notations.

Symbol	Description
$D = \{z_1, z_2, \dots, z_N\}$	Full dataset consisting of N data points.
z_i	i-th data point, may include features and label.
D^r	Subset of data to be removed.
θ	Model parameters.
$\bar{\theta}$	Optimal solution before data removal.
$\bar{\theta}_{-D^r}$	Optimal solution after removing data D^r.
$\eta = [\eta_1, \dots, \eta_N]$	Data weight vector.
η^r	Data weight vector after removing D^r (i.e., some entries are 0).
$\ell(z_i, \theta)$	Loss function for data point z_i.
$g_j(\cdot)$	j-th inequality constraint.
$h_t(\cdot)$	t-th equality constraint.
$\lambda = (\lambda_g, \lambda_h)$	Lagrange multipliers for constraints.
$\mathcal{L}(\eta, \theta, \lambda)$	Lagrangian of the weighted optimization problem.
$f(\eta, \theta, \lambda)$	Stationarity residual used in variational inequality (VI).
$N_E(\cdot)$	Normal cone to the feasible region E.
$\Delta\theta$, $\Delta\lambda$	Change in solution and multipliers due to data removal.
$\mathcal{N}(\cdot)$	PDE residual of the LWR model in PINNs.
$v(x,t)$	Observed average velocity.
$\hat{v}(x,t;\theta)$	Predicted velocity field by the PINN.
$S^r(x,t)$	Removed trajectory points in the spatiotemporal bin around (x,t).
$v(x,t;\eta)$	Weighted average velocity after down-weighting S^r.
$\phi(C,t)$	Penalty function applied to constraint violations.
$\mathbf{O}, \mathbf{A}$	Index sets for observed data and auxiliary PDE points.

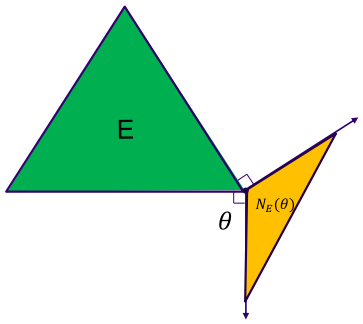


Fig. A.1. Example of Normal cone.

A.2. Normal Cone Definition

Let $E \subseteq \mathbb{R}^d$ be a closed convex set, and let $\theta \in E$. The normal cone to E at point θ, denoted by $N_E(x)$, is defined as (Dontchev and Rockafellar, 2009)

$$N_E(\theta) = \left\{ u \in \mathbb{R}^d \mid \langle u, \theta' - \theta \rangle \leq 0, \forall \theta' \in E \right\}.$$

This set contains all vectors that form non-acute angles with every feasible direction from θ within E. Intuitively, it characterizes the directions of non-descent at the boundary of the feasible set. An example is provided in Fig. A.1.

A.3. Optimality Condition of Auxiliary Problem

We present the derivation of the variational inequality (VI) optimality condition for the auxiliary problem (21).

Let $L(\eta_N, \theta, \lambda)$ denote the Lagrangian function defined in (17). We use L to denote the Lagrangian $L(\bar{\eta}, \bar{\theta}, \bar{\lambda})$ for brevity in Appendix A.3.

Following (11), the auxiliary VI of data weighted ITS model is:

$$\nabla_{\eta_N} f(\bar{\eta}, \bar{\theta}, \bar{\lambda}) \Delta\eta_N + \nabla_{(\theta,\lambda)} f(\bar{\eta}, \bar{\theta}, \bar{\lambda})[\Delta\theta; \Delta\lambda] + N_{\bar{E}}(\Delta\theta, \Delta\lambda) \ni \mathbf{0}, \tag{A.1}$$

where

$$f(\eta, \theta, \lambda) = \left(\nabla_\theta L, -\nabla_\lambda L\right)^\top \tag{A.2}$$

and

$$\bar{E} = \mathbb{R}^{\dim(\theta)} \times V,$$

$$V := \left\{ \Delta\lambda \in \mathbb{R}^{J+T-2} \middle| \begin{array}{ll} \Delta\lambda_g^j \geq 0 & \text{for } j \in [1, J-1] \text{ with } g_j(I_j, \bar{\theta}) = 0, \bar{\lambda}_g^j = 0, \\ \Delta\lambda_g^j = 0 & \text{for } j \in [1, J-1] \text{ with } g_j(I_i, \bar{\theta}) < 0. \end{array} \right\}. \tag{A.3}$$

The VI (A.1) can be translated to the requirements that:

$$\langle \nabla_\theta^2 L, \Delta\theta \rangle + \langle \nabla_{\theta\lambda} L, \Delta\lambda \rangle + \langle \nabla_{\theta\eta_N} L, \Delta\eta_N \rangle = 0, \tag{A.4}$$

where $\Delta\lambda_g^j \geq 0$ if $j \in I_0$ having $\bar{g}_j\left(\Delta\theta, z_{I_j}\right) = 0$. All other $\Delta\lambda_g^j$ are zero.

We now prove the optimality condition of auxiliary problem (21) is equivalent to VI (A.1).

We define the Lagrangian function of auxiliary problem (21):

$$\begin{aligned} L_{\text{aux}}(\Delta\theta, \gamma) =& L + \langle \nabla_\theta L, \Delta\theta \rangle + \frac{1}{2}\left\langle \Delta\theta,\ \nabla_{\theta\theta}^2 L \cdot \Delta\theta \right\rangle \\ &+ \left\langle \nabla_{\theta\eta_N}^2 L \cdot q,\ \Delta\theta \right\rangle + \sum_{j=1}^{J-1} \gamma_g^j g_j(z_{I_j}, \bar{\theta}) + \sum_{j=1}^{J-1} \gamma_g^j \left\langle \nabla_\theta g_j(z_{I_j}, \bar{\theta}),\ \Delta\theta \right\rangle + \sum_{t=1}^{T-1} \gamma_h^t \left\langle \nabla_\theta h_t(z_{I_t}, \bar{\theta}),\ \Delta\theta \right\rangle, \end{aligned} \tag{A.5}$$

where $\gamma = [\gamma_g^j, \gamma_h^t]$ are the Lagrangian multipliers. We have used the fact that $h_t(\bar{\theta}) = 0$, so constant terms in the constraint expansions vanish. The optimality condition of the auxiliary problem is:

$$\nabla_{\Delta\theta} L_{\text{aux}}(\Delta\theta, \gamma) = 0, \tag{A.6}$$

Note that $\nabla_\theta L = 0$ due to KKT of ITS model (13) and $\nabla_\lambda L = [g_j\left(z_{I_j}, \theta\right), h_t\left(z_{I_t}, \theta\right)]$. By substituting $q = \Delta\eta_N$ to (A.6), we have

$$\nabla_{\Delta\theta} L_{\text{aux}}\left(\Delta\theta, \gamma\right) = \left\langle \nabla_\theta^2 L, \Delta\theta \right\rangle + \nabla_{\theta\eta_N}^2 L \cdot q + \sum_{j=1}^{J-1} \gamma_g^j \nabla_\theta g_j\left(z_{I_j}, \bar{\theta}\right) + \sum_{t=1}^{T-1} \gamma_h^t \nabla_\theta h_t\left(z_{I_t}, \bar{\theta}\right) \tag{A.7}$$

$$= \left\langle \nabla_\theta^2 L, \Delta\theta \right\rangle + \langle \nabla_{\theta\lambda} L, \gamma \rangle + \left\langle \nabla_{\theta\eta_N} L, \Delta\eta_N \right\rangle. \tag{A.8}$$

The optimality condition (A.6) requires that:

$$\nabla_{\Delta\theta} L_{\text{aux}}(\Delta\theta, \gamma) = \left\langle \nabla_\theta^2 L, \Delta\theta \right\rangle + \langle \nabla_{\theta\lambda} L, \gamma \rangle + \left\langle \nabla_{\theta\eta_N} L, \Delta\eta_N \right\rangle = 0. \tag{A.9}$$

with $\gamma_g^j \geq 0$ when $j \in I_0$ having $\bar{g}_j\left(\Delta\theta, z_{I_j}\right) = 0$. All other γ_g^j are zero. This comes from the complementarity condition in the KKT conditions of the auxiliary problem.

It is straightforward to see that the solution $\Delta\theta$ to (A.4) is the same as the solution to (A.9). Thus, the auxiliary problem shares the same solution set as the auxiliary variational inequality. □

Appendix B. Constrained Machine Unlearning of SVM

This section analyzes the sensitivity of the SVM solution with the data weight η_N. As introduced in Section 5.1 , the data weighted SVM model is:

$$\begin{aligned} \min_{\theta=\{w,b,\xi\}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N-1}\xi_i + \eta_N \cdot C \max\left[0, 1 - y_N\left(w^\top x_N + b\right)\right] \\ \text{s.t} \quad & y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \\ & i = 1, \ldots, N-1. \end{aligned} \tag{Data Weighted SVM}$$

We use the parameterized Softplus function to replace the hinge loss $\max\left[0, 1 - y_N\left(w^\top x_N + b\right)\right]$. The parameterized Softplus function is defined as:

$$\text{Softplus}_\beta(1 - y_N\left(w^\top x_N + b\right)) = \frac{1}{\beta}\ln\left(1 + e^{\beta(1 - y_N(w^\top x_N + b))}\right), \tag{B.1}$$

where $\beta > 0$ controls the sharpness of the approximation. As $\beta \to \infty$, the Softplus function converges to the hinge loss, providing a smooth and differentiable approximation while maintaining numerical stability.

The Lagrangian function of the model (Data Weighted SVM) is

$$\begin{aligned} L(\eta_N, \theta, \alpha, \mu) = & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N-1}\xi_i + \frac{\eta_N C}{\beta}\ln\left(1 + e^{\beta(1 - y_N(w^\top x_N + b))}\right) \\ & + \sum_{i=1}^{N-1}\alpha_i\left[y_i\left(w^\top \mathbf{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{N-1}\mu_i\xi_i. \end{aligned} \tag{B.2}$$

Here, $\alpha = [\alpha_i]$ and $\mu = [\mu_i], i = 1, 2, \ldots, N-1$, are Lagrange multipliers corresponding to the first and second constraints, respectively. Based on the values of α_i, the training points are categorized as margin support vectors ($0 < \alpha_i < C$, set S), error support vectors ($\alpha_i = C$, set E), and reserve points ($\alpha_i = 0$, set R). The sets R and E can be further subdivided based on the relationship between $y_i\left(w^\top x_i + b\right)$ and the margin boundary:

$$R = \begin{cases} R_0 & \text{if } y_i\left(w^\top x_i + b\right) - 1 = 0, \\ R_1 & \text{if } y_i\left(w^\top x_i + b\right) - 1 < 0. \end{cases} \quad E = \begin{cases} E_0 & \text{if } y_i\left(w^\top x_i + b\right) - 1 = 0, \\ E_1 & \text{if } y_i\left(w^\top x_i + b\right) - 1 < 0. \end{cases} \tag{B.3}$$

When $\eta_N = \bar{\eta}_N = 1$, we denote the solution as $\bar{\theta} = \{\bar{w}, \bar{b}, \bar{\xi}\}$ and $\bar{\alpha}, \bar{\mu}$ as the optimal solution of the model (Data Weighted SVM). We assume the η_N moves along the direction of q. The auxiliary problem is given by:

$$\begin{aligned} \min_{\Delta\theta=(\Delta w,\Delta b,\Delta\xi)} \quad & \bar{g}_0(\Delta\theta) - \langle \gamma, \Delta\theta \rangle \\ \text{s.t.} \quad & g_i(\Delta\theta) := 1 - \xi_i - y_i\left(w^\top x_i + b\right) - y_i\left(\Delta w^\top x_i + \Delta b\right) - \Delta\xi_i \begin{cases} = 0 & \text{if } i \in S \cup E_0 \cup E_1, \\ \leq 0 & \text{if } i \in R_0, \\ \text{free} & \text{if } i \in R_1, \end{cases} \\ & h_i(\Delta\theta) := -\xi_i - \Delta\xi_i \begin{cases} = 0 & \text{if } i \in S \cup R_0 \cup R_1, \\ \leq 0 & \text{if } i \in E_0, \\ \text{free} & \text{if } i \in E_1. \end{cases} \\ & i = 1, 2, \ldots, N-1 \end{aligned} \tag{Auxiliary SVM}$$

Where $\bar{g}_0(\Delta\theta)$ is the second-order expansion of $L(\bar{\eta}_N, \bar{\theta}, \bar{\alpha}, \bar{\mu})$:

$$\begin{aligned} \bar{g}_0(\Delta\theta) &= L(\bar{\eta}_N, \bar{\theta}, \bar{\alpha}, \bar{\mu}) + \nabla_\theta L(\bar{\eta}_N, \bar{\theta}, \bar{\alpha}, \bar{\mu}) \cdot \Delta\theta + \frac{1}{2}\Delta\theta^\top \nabla_\theta^2 L(\bar{\eta}_N, \bar{\theta}, \bar{\alpha}, \bar{\mu})\Delta\theta \\ &= L(\bar{\eta}_N, \bar{\theta}, \bar{\alpha}, \bar{\mu}) + \frac{1}{2}\Delta w^\top \Delta w + \frac{\bar{\eta}_N C\beta}{2} M \cdot (\Delta w^\top x_N)^2 + C\bar{\eta}_N \beta \Delta b M \cdot (\Delta w^\top x_N) + C\bar{\eta}_N \beta M (\Delta b)^2, \end{aligned}$$

$$\gamma = \begin{pmatrix} qCy_N\sigma \cdot x_N \\ qC\sigma y_N \\ \mathbf{0} \in \mathbb{R}^{N-1} \end{pmatrix}, \tag{B.4}$$

$$\sigma = \frac{\exp\left(\beta(1 - y_N(\bar{w}^\top x_N + \bar{b})\right)}{1 + \ln\left(1 + \exp\left(\beta(1 - y_N(\bar{w}^\top x_N + \bar{b})\right)\right)}, \tag{B.5}$$

$$M = \sigma(1 - \sigma) \tag{B.6}$$

We denote the Lagrangian multipliers of the auxiliary problem (Auxiliary SVM) for g_i, h_i as z_i^g, z_i^h, respectively. Following (5), the first-order optimality condition of the auxiliary problem (Auxiliary SVM) w.r.t. θ, z_i^g, and z_i^h can be formulated as:

$$\begin{bmatrix} (I + wC\beta M \cdot x_N x_N^\top)\Delta w + wC\beta\Delta b M \cdot x_N - \sum_{i=1}^{N-1} z_i^g y_i x_i \\ wC\beta M \cdot \Delta w^\top x_N + wC\beta M\Delta b - \sum_{i=1}^{N-1} z_i^g y_i \\ -z_i^g - z_i^h \quad (i = 1, 2, \ldots, N-1) \end{bmatrix} - v = 0, \tag{B.7}$$

The change of w, denoted by Δw, can be obtained by rearranging (B.7):

$$\Delta w = \sum_{i=1}^{N-1} z_i^g y_i \left(\mathbf{x}_i - \mathbf{x}_N\right) \tag{B.8}$$

The values of z_i^g can be derived by solving the auxiliary problem (Auxiliary SVM). Using the complementary slackness condition of the auxiliary model, we can provide the following value ranges:

$$z_i^g = \begin{cases} 0, & \text{if } i \in R_1 \cup E_1 \\ \geq 0, & \text{for } i \in R_0 \text{ such that } -y_i\left(\mathbf{x}_i^\top \Delta\mathbf{w} + \Delta b\right) - \Delta\xi_i = 0 \\ 0, & \text{for } i \in R_0 \text{ such that } -y_i\left(\mathbf{x}_i^\top \Delta\mathbf{w} + \Delta b\right) - \Delta\xi_i < 0 \\ free & \text{for } i \in S \cup E_0 \end{cases} \tag{B.9}$$

After identifying Δw, the value of Δb can be derived from the following equation:

$$y_i\left(\mathbf{x}_i^\top \Delta\mathbf{w}\right) + y_i\Delta b + \Delta\xi_i = 0, \quad \text{for any } i \in S \tag{B.10}$$

By solving the auxiliary problem (Auxiliary SVM), we can derive $\Delta\theta = \{\Delta w, \Delta b, \Delta\xi\}$ without retraining the SVM.

Appendix C. Multiple Data Points Deletion Scenario with Multi-Constraint Involvement

We consider the removal of a set of data points $D^r = [z_{i_1}, z_{i_2}, \ldots, z_{i_m}]$. The corresponding gold standard solution $\bar{\theta}_{-D^r}$ is obtained by retraining the ITS model (13) after excluding D^r from both the objective and the constraints. Let J_R (resp. T_R) denote the indices of inequality (resp. equality) constraints involving any $z_i \in D^r$. The corresponding gold standard ITS model is formulated as:

$$\begin{aligned} \bar{\theta}_{-D^r} = \arg\min_{\theta} & \sum_{z_i \notin D^r} \ell(z_i, \theta) \\ \text{s.t.} \quad & g_j(z_{I_j} \setminus D^r, \theta) \leq 0, \quad \forall j \in J_R, \\ & g_j(z_{I_j}, \theta) \leq 0, \quad \forall j \in \{1, \ldots, J\} \setminus J_R, \\ & h_t(z_{I_t} \setminus D^r, \theta) = 0, \quad \forall t \in T_R, \\ & h_t(z_{I_t}, \theta) = 0, \quad \forall t \in \{1, \ldots, T\} \setminus T_R. \end{aligned} \tag{C.1}$$

To approximate the gold standard solution $\bar{\theta}_{-D^r}$ without retraining, we adopt a data-weighted formulation by introducing a weight $\eta_i \in [0, 1]$ for each $z_i \in D^r$. Define the vector $\eta = [\eta_{i_1}, \eta_{i_2}, \ldots, \eta_{i_m}]$. When all weights are 1, the original ML-ITS model is recovered; when all are 0, the points in D^r are removed. Following the model (16), we have a data weighted ITS formulated as:

$$\begin{aligned} \theta(\eta) = \arg\min_{\theta} & \sum_{z_i \notin D^r} \ell(z_i, \theta) + \sum_{z_i \in D^r} \eta_i \cdot \ell(z_i, \theta) \\ & + \sum_{j \in J_R} \phi\Big(C_g, g_j(z_{I_j} \setminus D^r, \eta \cdot D^r, \theta)\Big) \\ & + \sum_{t \in T_R} \{\phi\Big(C_h, h_t(z_{I_t} \setminus D^r, \eta \cdot D^r, \theta)\Big) + \phi\Big(C_h, -h_t(z_{I_t} \setminus D^r, \eta \cdot D^r, \theta)\Big)\} \\ \text{s.t.} \quad & g_j(z_{I_j}, \theta) \leq 0, \quad \forall j \in \{1, \ldots, J\} \setminus J_R, \\ & h_t(z_{I_t}, \theta) = 0, \quad \forall t \in \{1, \ldots, T\} \setminus T_R. \end{aligned} \tag{C.2}$$

Let $L(\eta, \theta, \lambda)$ denote the Lagrangian:

$$\begin{aligned} L(\eta, \theta, \lambda) = & \sum_{z_i \notin D^r} \ell(z_i, \theta) + \sum_{z_i \in D^r} \eta_i \cdot \ell(z_i, \theta) \\ & + \sum_{j \in J_R} \phi\Big(C_g, g_j(z_{I_j} \setminus D^r, \eta \cdot D^r, \theta)\Big) + \sum_{t \in T_R} \phi\Big(C_h, h_t(z_{I_t} \setminus D^r, \eta \cdot D^r, \theta)\Big) \\ & + \sum_{t \in T_R} \phi\Big(C_h, -h_t(z_{I_t} \setminus D^r, \eta \cdot D^r, \theta)\Big) + \sum_{j \in \{1,\dots,J\} \setminus J_R} \lambda_g^j g_j(z_{I_j}, \theta) + \sum_{t \in \{1,\dots,T\} \setminus T_R} \lambda_h^t h_t(z_{I_t}, \theta). \end{aligned} \tag{C.3}$$

We define $\bar{\theta} = \theta(\eta = \mathbf{1})$, $\bar{\lambda}$ as its Lagrange multiplier, and define $q = -\mathbf{1} \in \mathbb{R}^m$ as the perturbation direction for η from $\bar{\eta} = 1$. Then the auxiliary problem solves for $\Delta\theta$:

$$\begin{aligned} \min_{\Delta\theta} \quad & \bar{g}_0(\Delta\theta) + \Big\langle \nabla^2_{\theta\eta} L(\bar{\eta}, \bar{\theta}, \bar{\lambda}) \cdot \boldsymbol{q}, \Delta\theta \Big\rangle \\ \text{s.t.} \quad & \bar{g}_j(\Delta\theta, z_{I_j}) = 0, \quad \forall j \in \tilde{I} \setminus \tilde{I}_0 \\ & \bar{g}_j(\Delta\theta, z_{I_j}) \le 0, \quad \forall j \in \tilde{I}_0 \\ & \bar{h}_t(\Delta\theta, z_{I_t}) = 0, \quad \forall t \in \{1, \dots, T\} \setminus T_R \end{aligned} \tag{C.4}$$

Here, $\bar{g}_0(\Delta\theta)$ is the second-order expansion of the Lagrangian function:

$$\bar{g}_0(\Delta\theta) = L(\bar{\eta}, \bar{\theta}, \bar{\lambda}) + \Big\langle \nabla_\theta L(\bar{\eta}, \bar{\theta}, \bar{\lambda}), \Delta\theta \Big\rangle + \frac{1}{2} \Delta\theta^\top \nabla^2_{\theta\theta} L(\bar{\eta}, \bar{\theta}, \bar{\lambda}) \Delta\theta \tag{C.5}$$

The linearized constraints are:

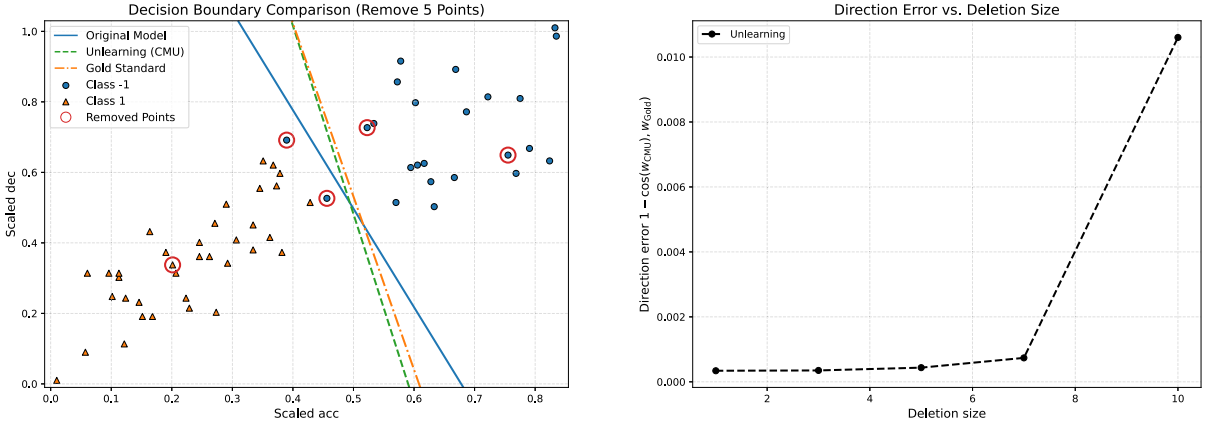
$$\begin{aligned} \bar{g}_j(\Delta\theta, z_{I_j}) &= g_j(z_{I_j}, \bar{\theta}) + \Big\langle \nabla_\theta g_j(z_{I_j}, \bar{\theta}), \Delta\theta \Big\rangle \\ \bar{h}_t(\Delta\theta, z_{I_t}) &= h_t(z_{I_t}, \bar{\theta}) + \Big\langle \nabla_\theta h_t(z_{I_t}, \bar{\theta}), \Delta\theta \Big\rangle \end{aligned} \tag{C.6}$$

We define the index sets as follows:

$$\begin{aligned} \tilde{I} &= \Big\{ j \in \{1, \dots, J\} \setminus J_R \mid g_j(\bar{\theta}, z_{I_j}) = 0 \Big\} \\ \tilde{I}_0 &= \Big\{ j\{1, \dots, J\} \setminus J_R \mid g_j(\bar{\theta}, z_{I_j}) = 0, \bar{\lambda}_g^j = 0 \Big\} \subset \tilde{I} \end{aligned} \tag{C.7}$$

By solving the auxiliary problem (C.4), which is also a quadratic program, we obtain a local linear approximation to the solution change when removing multiple data points D^r.

Appendix D. Constrained Machine Unlearning of SVM with Multiple Data Points Deletion



(a) Decision boundary comparison after removing five samples.

(b) Direction error versus deletion size.

Fig. D.1. Comparison of multi-point unlearning. (Left) Decision boundaries of the original model, CMU-based unlearning, and retraining after removing five samples. (Right) Direction error between the unlearned and retrained models for varying deletion sizes.

We study the effect of removing multiple training samples in a binary classification task in Section 5.1 with a smooth hinge loss. We consider different deletion sizes $\{1, 3, 5, 7, 10\}$ from a dataset of 60 samples and compare the CMU-based unlearning approximation with the gold-standard retraining after data removal.

Fig. D.1(a) illustrates the decision boundary before and after removing five samples. Even after removing five samples, the CMU method still produces an approximation that is close to the retrained solution. Fig. D.1(b) shows the direction error as a function

of the deletion size. We define the direction error as $1 - \cos(w_{\text{CMU}}, w_{\text{Gold}})$, which measures the difference in the orientation of the decision boundaries between the unlearned model w_{CMU} and the retrained model w_{Gold}.

We observe that the error remains very small when the deletion size is small (e.g., deletion size ≤ 5 or less than 10% of the entire dataset), indicating that the CMU approximation is accurate in the local regime. As the deletion size increases, the error becomes more pronounced, particularly when the deletion size reaches 10, suggesting that the approximation deteriorates for larger perturbations. Interestingly, the error is not strictly monotonic with respect to the deletion size. For instance, the error at deletion size 5 is nearly zero, while it increases significantly at deletion size 10. This non-monotonic behavior indicates that the approximation error is not solely determined by the number of removed samples. Instead, it depends on the joint effect of the removed samples on the model. Since each sample contributes a vector-valued perturbation to the model parameters, their combined effect may partially cancel out, resulting in a small deviation even when multiple samples are removed. However, when the removal leads to a more substantial shift in the solution, the approximation error increases noticeably. The experiment implies that CMU (or machine unlearning in general) may work well if a small portion (less than 10%) of the training data is to be removed, which should work appropriately for data deletion requests due to privacy protection or data poisoning.

E. Data Weight Decay Approach for Machine Unlearning

We introduce a data weight decay approach when the data size for deletion is larger (e.g., larger than 10%). Instead of directly removing a data point by changing its weight from 1 to 0 in a single step, the approach partitions this change into several smaller steps. In our implementation, the interval from 1 to 0 is divided into 3 to 5 sub-intervals.

At each step, we perform sensitivity analysis to estimate how the model solution changes under a small weight perturbation, and then update the solution accordingly. This updated solution is used as the reference point for the next step. This procedure is repeated until the weights of the data to be removed are reduced to 0. The final model obtained through this gradual procedure is called the **weight-decay unlearned model**. By contrast, the baseline approach that directly changes the data weights from 1 to 0 in one step is referred to as the **one-shot unlearned model**.

We conduct experiments on the SVM model with 9 data points removed. The SVM results are shown in Fig. E.1a. Compared with the one-shot unlearning update, the proposed weight-decay approach produces a decision boundary that is visibly closer to the gold-standard retrained model. This improvement is also confirmed quantitatively by the parameter distance to the gold standard model solution:

$$\left\|\theta_{\text{one-shot}} - \theta_{\text{gold}}\right\| = 1.684270, \quad \left\|\theta_{\text{weight-decay}} - \theta_{\text{gold}}\right\| = 0.610956.$$

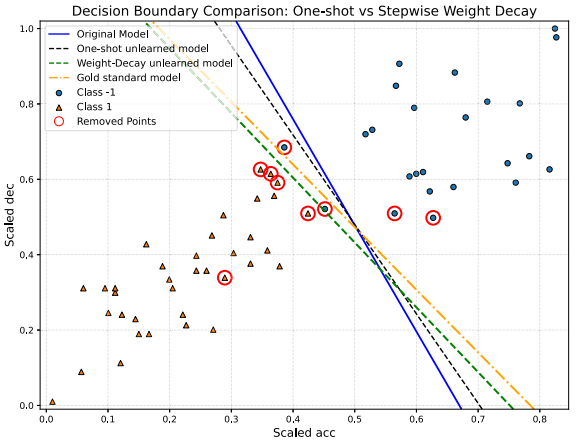
This trend is also supported by Fig. E.1b, where the weight-decay approach consistently achieves smaller direction error than the one-shot update across different deletion sizes, especially when the number of removed points becomes larger. As in Appendix D, we define the direction error as $1 - \cos\left(w_{\text{CMU}}, w_{\text{Gold}}\right)$, which quantifies the discrepancy in the orientation of the decision boundary between the unlearned model w_{CMU} and the gold-standard retrained model w_{Gold}.

Table E.1
Weight Decay Unlearned Model Performance on PINN (15% data removal.)

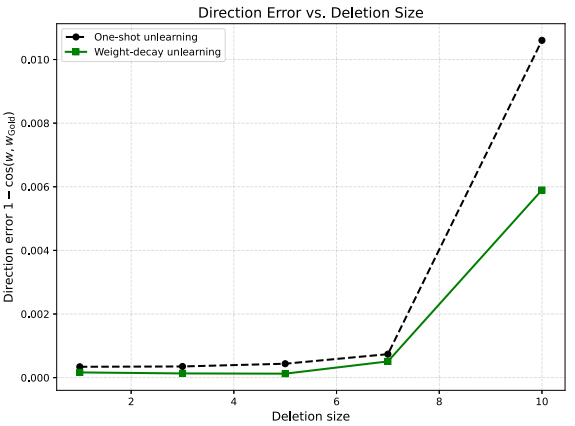
Model	MAE (Data Loss)	MAE (Physics Loss)	Relative L_2 Error (%)
Original Model	1.43	2.1×10^{-2}	8.46
Retrained Model	2.46	1.5×10^{-2}	14.49
One-shot unlearned model	3.94	1.9×10^{-2}	22.06
Weight-decay unlearned model	2.67	1.5×10^{-2}	16.06

We further evaluate the proposed weight-decay unlearning strategy on the PINN model. The quantitative results are reported in Table E.1, and the predicted velocity fields are shown in Fig. E.2. Compared with the one-shot unlearned model, the weight-decay unlearned model is consistently closer to the retrained model across all three metrics. In particular, its data loss is reduced from 3.94 to 2.67, much closer to the retrained model value of 2.46. Its physics loss, 1.5×10^{-2}, matches that of the retrained model exactly, while the one-shot unlearned model has a larger physics loss of 1.9×10^{-2}. Moreover, the relative L_2 error is improved substantially from 22.06% for the one-shot unlearned model to 16.06% for the weight-decay unlearned model, which is close to the retrained model error of 14.49%. The qualitative comparison in Fig. E.2 shows a similar trend. The intermediate states in the top row illustrate that the weight decay procedure updates the solution smoothly as the weights of data to be removed are gradually reduced. These results show that the weight decay update tracks the retraining solution substantially better than the one-shot update. In particular, gradually reducing the weights allows the local sensitivity approximation to remain more accurate at each stage, leading to a final solution that is much closer to the gold-standard retrained model.

However, the downside of the weight-decay method is that it is computationally more demanding than the one-shot method and requires multiple one-shot unlearning depending on how many gradual steps are performed (usually 3-5 steps). Therefore, the analysis and results here are merely for testing the extreme scenarios where a large size of data needs to be deleted and understanding how the weight-decay method may perform. For practical applications, a full retraining may be required if large size of training data needs to be removed.

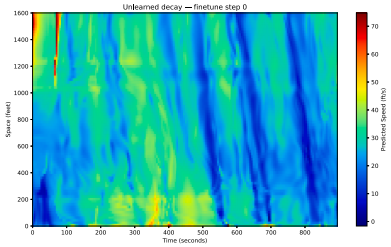


(a) Decision boundary comparison among the original model, one-shot unlearning, weight-decay unlearning, and gold standard model.

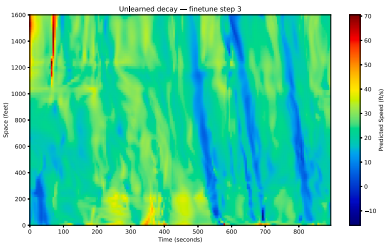


(b) Direction error versus deletion size for one-shot unlearning and weight-decay unlearning.

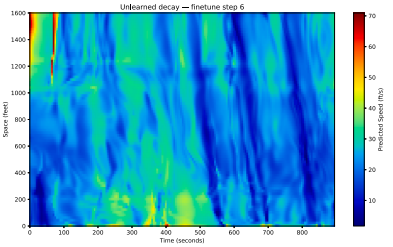
Fig. E.1. Comparison of unlearning performance in the SVM experiment.



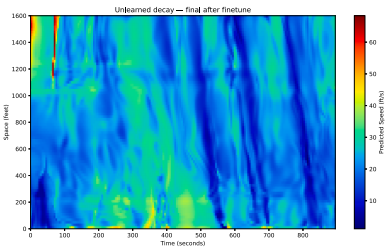
(a) Intermediate State (weight=0.9)



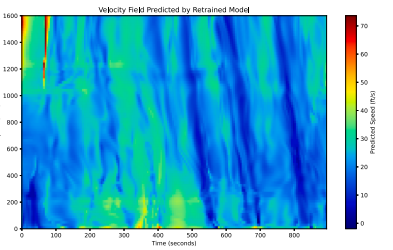
(b) Intermediate State (weight=0.6)



(c) Intermediate State (weight=0.3)



(d) weight-decay unlearned model



(e) Retrained model

Fig. E.2. Comparison of velocity fields under different unlearning strategies. The top row shows intermediate states of the decay-based unlearning process at different steps. The bottom row compares the final decay-based unlearned result (weight = 0) with the retrained model.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L., 2016. Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 308–318.

Ban, X.J., Hao, P., Sun, Z., 2011. Real time queue length estimation for signalized intersections using travel times from mobile sensors. Transportation Research Part C: Emerging Technologies 19 (6), 1133–1156.

Barhoumi, O., Zaki, M.H., Tahar, S., 2025. Formally constrained reinforcement learning for traffic signal control at intersections. In: 2025 IEEE International systems Conference (SysCon). IEEE, pp. 1–8.

Birattari, M., Bontempi, G., Bersini, H., 1998. Lazy learning meets the recursive least squares algorithm. Advances in neural information processing systems 11.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C.A., Jia, H., Travers, A., Zhang, B., Lie, D., Papernot, N., 2021. Machine unlearning. In: 2021 IEEE Symposium on Security and Privacy (SP). IEEE, pp. 141–159.

Cao, Y., Yang, J., 2015. Towards making systems forget with machine unlearning. In: 2015 IEEE symposium on security and privacy. IEEE, pp. 463–480.

Cui, Z., Henrickson, K., Ke, R., Wang, Y., 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. IEEE Transactions on Intelligent Transportation Systems 21 (11), 4883–4894.

De Montjoye, Y.-A., Hidalgo, C.A., Verleysen, M., Blondel, V.D., 2013. Unique in the crowd: The privacy bounds of human mobility. Scientific reports 3 (1), 1–5.

Dontchev, A.L., Rockafellar, R.T., 2009. Implicit functions and solution mappings. Vol. 543. Springer.

Facchinei, F., Pang, J.-S., 2003. Finite-dimensional variational inequalities and complementarity problems. Springer.
Feng, J., Rong, C., Sun, F., Guo, D., Li, Y., 2020. Pmf: A privacy-preserving human mobility prediction framework via federated learning. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 4 (1), 1–21.
Firdaus, M., Larasati, H.T., Hyune-Rhee, K., 2025. Blockchain-based federated learning with homomorphic encryption for privacy-preserving healthcare data sharing. Internet of Things 31, 101579.
Geng, M., Li, J., Xia, Y., Chen, X.M., 2023. A physics-informed transformer model for vehicle trajectory prediction on highways. Transportation research part C: emerging technologies 154, 104272.
Gfrerer, H., Outrata, J.V., 2016. On lipschitzian properties of implicit multifunctions. SIAM Journal on Optimization 26 (4), 2160–2189.
Golatkar, A., Achille, A., Soatto, S., 2020. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16. Springer, pp. 383–398.
Gong, T., Zhu, L., Yu, F.R., Tang, T., 2023. Edge intelligence in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems 24 (9), 8919–8944.
Hasan, M., Mohan, S., Shimizu, T., Lu, H., 2020. Securing vehicle-to-everything (v2x) communication platforms. IEEE Transactions on Intelligent Vehicles 5 (4), 693–713.
Haydari, A., Yılmaz, Y., 2020. Deep reinforcement learning for intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems 23 (1), 11–32.
Huang, A.J., Agarwal, S., 2022. Physics-informed deep learning for traffic state estimation: Illustrations with LWR and CTM models. IEEE Open Journal of Intelligent Transportation Systems 3, 503–518.
Iqbal, M.U., Lim, S., 2010. Privacy implications of automated GPS tracking and profiling. IEEE technology and society magazine 29 (2), 39–46.
Koh, P.W., Liang, P., 2017. Understanding black-box predictions via influence functions. In: International conference on machine learning. PMLR, pp. 1885–1894.
Krantz, S.G., Parks, H.R., 2002. The implicit function theorem: history, theory, and applications. Springer Science & Business Media.
Law, J., 1986. Robust statistics–the approach based on influence functions.
Li, W., Ban, X.J., Zheng, J., Liu, H.X., Gong, C., Li, Y., 2020. Real-time movement-based traffic volume prediction at signalized intersections. Journal of Transportation Engineering, Part A: Systems 146 (8), 04020081.
Lu, J., Li, C., Wu, X.B., Zhou, X.S., 2023. Physics-informed neural networks for integrated traffic state and queue profile estimation: A differentiable programming approach on layered computational graphs. Transportation Research Part C: Emerging Technologies 153, 104224.
Luo, Z.-Q., Pang, J.-S., Ralph, D., 1996. Mathematical programs with equilibrium constraints. Cambridge University Press.
Nasr, M., Shokri, R., Houmansadr, A., 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE symposium on security and privacy (SP). IEEE, pp. 739–753.
Nguyen, T.T., Huynh, T.T., Ren, Z., Nguyen, P.L., Liew, A. W.-C., Yin, H., Nguyen, Q. V.H., 2022. A survey of machine unlearning. arXiv preprint arXiv:2209.02299.
Qi, T., Wu, F., Wu, C., Lyu, L., Xu, T., Liao, H., Yang, Z., Huang, Y., Xie, X., 2022. Fairvfl: A fair vertical federated learning framework with contrastive adversarial learning. Advances in neural information processing systems 35, 7852–7865.
Rao, B., Zhang, J., Wu, D., Zhu, C., Sun, X., Chen, B., 2024. Privacy inference attack and defense in centralized and federated learning: A comprehensive survey. IEEE Transactions on Artificial Intelligence .
Regulation, P., 2016. Regulation (EU) 2016/679 of the european parliament and of the council. Regulation (eu) 679, 2016.
Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., Savarese, S., 2019. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1349–1358.
Shi, R., Mo, Z., Di, X., 2021. 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. Vol. 35, pp. 540–547.
Spingarn, J.E., 1978. Generic conditions for optimality in constrained minimization problems .
Spingarn, J.E., Rockafellar, R.T., 1979. The generic nature of optimality conditions in nonlinear programming. Mathematics of Operations Research 4 (4), 425–430.
Sun, Z., Ban, X.J., 2013. Vehicle classification using GPS data. Transportation Research Part C: Emerging Technologies 37, 102–117.
Sun, Z., Zan, B., Ban, X.J., Gruteser, M., 2013. Privacy protection method for fine-grained urban traffic modeling using mobile sensors. Transportation Research Part B: Methodological 56, 50–69.
Thodi, B.T., Khan, Z.S., Jabari, S.E., Menendez, M., 2022. Incorporating kinematic wave theory into a deep learning method for high-resolution traffic speed estimation. IEEE Transactions on Intelligent Transportation Systems 23 (10), 17849–17862.
Triantafillou, E., Kairouz, P., Pedregosa, F., Hayes, J., Kurmanji, M., Zhao, K., Dumoulin, V., Junior, J.J., Mitliagkas, I., Wan, J., et al., 2024. Are we making progress in unlearning? findings from the first neurIPS unlearning competition. arXiv preprint arXiv:2406.09073.
Truex, S., Liu, L., Gursoy, M.E., Yu, L., Wei, W., 2018. Towards demystifying membership inference attacks. arXiv preprint arXiv:1807.09173.
Veres, M., Moussa, M., 2019. Deep learning for intelligent transportation systems: A survey of emerging trends. IEEE Transactions on Intelligent transportation systems 21 (8), 3152–3168.
Wang, F., Hong, Y., Ban, X., 2023. Infrastructure-enabled GPS spoofing detection and correction. IEEE Transactions on Intelligent Transportation Systems 24 (12), 13878–13892.
Wang, F., Wang, X., Ban, X.J., 2024a. Data poisoning attacks in intelligent transportation systems: A survey. Transportation Research Part C: Emerging Technologies 165, 104750.
Wang, F., Wang, X., Hong, Y., Rockafellar, R.T., Ban, X.J., 2024b. Data poisoning attacks on traffic state estimation and prediction. Transportation Research Part C: Emerging Technologies , 104577.
Wang, Q., Yang, K., 2024. Privacy-preserving data fusion for traffic state estimation: A vertical federated learning approach. Transportation Research Part C: Emerging Technologies 168, 104743.
Wang, X., Wang, F., Ban, X.J., 2025. Set-valued sensitivity analysis of deep neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39, pp. 21304–21311.
Wang, X., Wang, F., Hong, Y., Ban, X., 2024c. Transferability in data poisoning attacks on spatiotemporal traffic forecasting models. Available at SSRN 4827065 .
Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H., 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE conference on computer communications. IEEE, pp. 2512–2520.
Wu, Y., Dobriban, E., Davidson, S., 2020. Deltagrad: Rapid retraining of machine learning models. In: International Conference on Machine Learning. PMLR, pp. 10355–10366.
Xia, J., Yang, Y., Shen, J., Wang, S., Cao, J., 2025. FairTP: A prolonged fairness framework for traffic prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39, pp. 26391–26399.
Xu, H., Yuan, J., Li, W., Zhang, F., Zlatanova, S., Ban, X., Pettit, C., Ye, X., 2025. Next-generation intelligent transportation systems using multimodal generative AI and diverse modalities of traffic data. Available at SSRN 5547519.
Ying, Z., Cao, S., Liu, X., Ma, Z., Ma, J., Deng, R.H., 2022. Privacysignal: Privacy-preserving traffic signal control for intelligent transportation system. IEEE Transactions on Intelligent Transportation Systems 23 (9), 16290–16303.
Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., Chen, C., 2011. Data-driven intelligent transportation systems: A survey. IEEE transactions on intelligent transportation systems 12 (4), 1624–1639.
Zhou, J., Yang, K., 2024. A parameter privacy-preserving strategy for mixed-autonomy platoon control. Transportation Research Part C: Emerging Technologies 169, 104885.
Zhu, L., Liu, Z., Han, S., 2019. Deep leakage from gradients. Advances in neural information processing systems 32.