Toward Robust Decentralized Learning and Autonomous Perception

by

Ziba Parsons

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer and Information Science)
in The University of Michigan-Dearborn
2026

Doctoral Committee:

Assistant Professor Ang Li, Chair
Associate Professor Birhanu Eshete
Associate Professor Jinhu Hong
Professor Di Ma

Ziba Parsons

zibapars@umich.edu

ORCID iD: 0000-0002-1211-0595

https://orcid.org/0000-0002-1211-0595

© Ziba Parsons 2026

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES vii

LIST OF APPENDICES viii

ABSTRACT ix

CHAPTER

1. Introduction 1

1.1 Background 2
1.2 Research Problems 6
1.3 Limitations of Existing Solutions 7
1.4 Our Solutions 9
1.5 Research Questions and Objectives 11
1.6 Dissertation Overview and Structure 12

2. Related Work 14

2.1 Federated Learning and Data Heterogeneity 14
2.2 ADMM-Based and Optimization-Driven FL 16
2.3 LiDAR-Based 3D Object Detection 18
2.4 Security and Backdoor Attacks in 3D Object Detection 19

3. Reliable Personalized Federated Learning 23

3.1 Problem Formulation 24
3.2 Random Walk Stochastic ADMM (RWSADMM) 26
3.3 Theoretical Analysis 29
3.4 Experimental Evaluation 32
3.4.1 Models 33
3.4.2 Graph Construction 34
3.4.3 Hyperparameter Tuning 34

3.4.4 CIFAR-10 Results . . . 36
3.4.5 Synthetic Results . . . 36
3.4.6 Different Number of Clients . . . 37
3.5 Discussion . . . 39
3.5.1 Limitations . . . 39
3.5.2 Future Work . . . 39
3.6 Conclusion . . . 40

4. Security of LiDAR-Based 3D Object Detection . . . 41

4.1 Threat Model . . . 42
4.2 Methodology . . . 43
4.2.1 Surrogate Model . . . 44
4.2.2 Trigger Optimization . . . 44
4.2.3 Trigger Deployment . . . 48
4.3 Experimental Results . . . 51
4.3.1 General Setup . . . 51
4.3.2 Performance Evaluation . . . 54
4.4 Object Disappearance . . . 60
4.4.1 Misclassification-Aligned Trigger . . . 61
4.4.2 Disappearance-Aligned Trigger: The Occlusion Baseline 66
4.4.3 Dual-Objective Trigger . . . 68
4.4.4 Placement and Physical Realizability . . . 71
4.4.5 Cross-Architecture Transfer: the SECOND Victim . . . 73
4.5 Defenses . . . 82
4.5.1 Training-Time Data Curation . . . 82
4.5.2 Inference-Time Input Sanitization . . . 83
4.5.3 Synthesis . . . 84
4.6 Discussion . . . 85
4.6.1 Physical Realizability . . . 85
4.6.2 Limitations . . . 86
4.6.3 Future Work . . . 87
4.7 Conclusion . . . 88

5. Unifying Discussions and Future Directions . . . 89

5.1 Unifying Perspective . . . 89
5.2 From Reliable Decentralized Training to a Trusted Perception Model 90
5.3 Security as the Other Face of Decentralized Data . . . 91
5.4 Implications for Autonomous Driving Safety . . . 91
5.4.1 Vulnerability of Current Pipelines . . . 91
5.4.2 Need for Security-Aware Training . . . 92
5.5 Future Directions . . . 92
5.6 Conclusion . . . 93

APPENDICES 95
A.1 RWSADMM: Convergence Analysis 96
B.1 RWSADMM: Convergence Rate Analysis 109

BIBLIOGRAPHY 111

LIST OF FIGURES

Figure

1.1	Federated learning: one round of Federated Averaging	3
1.2	Camera versus LiDAR on a KITTI frame	5
1.3	The three stages of a backdoor attack	6
1.4	RWSADMM: mobile-server training along a random walk	9
1.5	Overview of the clean-label LiDAR backdoor attack	11
3.1	Convergence on MNIST across MLR, MLP, and CNN models	33
3.2	Effect of β on RWSADMM convergence (MNIST)	35
3.3	Effect of κ on RWSADMM convergence (MNIST)	36
3.4	Convergence on CIFAR-10 across MLR, MLP, and CNN models	37
3.5	Convergence on the Synthetic dataset for MLR and MLP	38
3.6	RWSADMM convergence on MNIST with 20, 50, and 100 clients	38
4.1	The MIRAGE clean-label backdoor pipeline	46
4.2	The PointPillars detection architecture	52
4.3	KITTI point-cloud and camera views of the three evaluated classes	53
4.4	Poison-rate, radius, and density sweep of misclassification success	55
4.5	Random versus optimized trigger on a representative pedestrian	56
4.6	Disappearance rate over the poison-rate, radius, and density sweep	58
4.7	Clean-utility cost over the poison-rate, radius, and density sweep	59
4.8	Stealth–strength trade-off across the full sweep	60
4.9	Car-proportioned box versus isotropic sphere trigger geometry	61
4.10	Inference-time evasion versus blob radius on a clean victim	69
4.11	Blob radius and poison rate sweep for the dual-objective trigger	71
4.12	Blob radius and poison rate sweep for the pure-disappearance trigger on SECOND	80
4.13	The disappearance trigger as a buildable physical object	87

LIST OF TABLES

Table

1.1	Backdoor attacks against LiDAR 3DOD	8
3.1	Accuracy and time comparison on MNIST, Synthetic, and CIFAR-10	34
3.2	Accuracy and time for 20, 50, and 100 clients	39
4.1	Targeted misclassification across victim detectors on KITTI *val*	53
4.2	Training-time poisoning: cyclist disappearance by radius and poison rate	64
4.3	Optimized phantom versus random blob under training-time poisoning	65
4.4	Inference-time physical evasion by blob radius and placement	68
4.5	Training-time poisoning is counterproductive for disappearance	70
4.6	Dual-objective trigger by misclassification weight λ	70
4.7	Deployment-placement sweep for the misclassification-aligned trigger	72
4.8	Deployable pannier trigger versus the idealized floating sphere	73
4.9	Robustness of the disappearance to sensor-realistic ranging noise	73
4.10	Cross-architecture transfer of the disappearance attack to SECOND	74
4.11	Misclassification-aligned phantom trigger on SECOND	78
4.12	Optimized phantom versus matched random blob on SECOND	79

LIST OF APPENDICES

Appendix

A. RWSADMM Convergence Analysis 96

B. RWSADMM Convergence Rate Analysis 109

ABSTRACT

Modern machine learning systems are increasingly deployed in decentralized, resource-constrained, and safety-critical environments, such as autonomous driving and edge-based intelligent systems. In these settings, models must operate reliably under heterogeneous data distributions, limited communication infrastructure, and heightened security risks. This dissertation investigates the reliability and security of learning systems in such environments, with a particular focus on federated learning and autonomous driving perception.

The first part of this dissertation addresses the challenge of reliable federated learning under extreme heterogeneity and intermittent connectivity. We study infrastructure-less federated learning in which a mobile server communicates opportunistically with distributed clients. To this end, we propose Random Walk Stochastic Alternating Direction Method of Multipliers (RWSADMM), a personalized federated learning framework that combines server mobility with a stochastic ADMM-based optimization scheme. By enforcing hard inequality constraints among neighboring clients rather than global consensus, the proposed method achieves personalization while reducing communication and computation overhead. We provide theoretical convergence guarantees and demonstrate empirical improvements over state-of-the-art methods under non-IID data distributions.

The second part turns from the *reliability* of learning on decentralized data to the *security* of the models it produces, examining Light Detection and Ranging (LiDAR)-based 3D object detection systems. The same reliance on data aggregated from many uncontrolled sources that motivates federated learning also exposes safety-critical perception models to training-time manipulation, making their adversarial vulnerabilities essential to understand. We investigate clean-label backdoor attacks against LiDAR-based 3D detectors, in which physically plausible point-cloud triggers are injected during training without changing ground-truth labels. The attacker operates under a stringent *scene-only* threat model—never submitting data, altering labels, or accessing the victim model, but only placing an inconspicuous physical object in the data-collection environment. Using the KITTI dataset and widely deployed detection architectures, we demonstrate that such attacks can reliably induce both targeted misclassification and object disappearance—suppressing a real object

so the detector reports nothing where one exists—while largely preserving clean detection performance, revealing critical vulnerabilities in current perception pipelines.

Taken together, the two parts trace a single arc—from learning *reliably* on decentralized data to *securing* the models it yields—and show how reliability and security are coupled in decentralized and autonomous learning systems. The dissertation contributes algorithmic, theoretical, and security-focused insights for deploying machine learning in real-world, safety-critical environments where the data cannot be fully controlled.

CHAPTER 1

Introduction

Modern machine learning is increasingly trained not in a controlled data center but at fleet scale, on data streamed from many devices that no single party owns or curates. Autonomous driving is the setting where this shift matters most: the perception models that steer a vehicle are trained on point clouds and images aggregated from large fleets of data-collection vehicles, public benchmarks, and crowd-sourced logs, and must then be trusted in a safety-critical control loop. A model built this way must be not only accurate but *trustworthy*, and trustworthiness here rests on two coupled requirements. *Reliability* concerns whether the model can be trained well in the first place, when its data are distributed across many heterogeneous, intermittently connected sources. *Security* concerns whether the trained model can be trusted when some of the parties supplying its data cannot be. This dissertation examines these two requirements through two research projects that sit at opposite ends of a single arc. The first asks whether federated learning (FL) can remain *reliable* under the opportunistic connectivity and extreme data heterogeneity that arise whenever training is spread across many uncontrolled sources—of which a vehicle fleet is a canonical instance. The second asks whether the perception models that such open, decentralized pipelines produce can be *secured*, through clean-label backdoor attacks on Light Detection and Ranging (LiDAR)-based 3D object detection (3DOD). What links them is the pipeline itself—a single shift away from centrally curated data toward training data pooled from many partially trusted sources. Reliability and security are the two faces of that shift: the first project confronts the *benign* corruption of the training distribution by statistical heterogeneity, the second its *adversarial* corruption by a hidden trigger. For each project we move from background to the research problems it raises, the limitations of existing solutions, and finally our solution, returning in the closing sections to the reliability–security thread that unites them.

1.1 Background

Federated Learning (FL) [1–4] is a distributed machine learning paradigm that enables clients to learn a shared model without sharing their private data. Unlike traditional machine learning approaches that rely on central servers for model training, FL allows clients to collaborate and train the model in a distributed manner, overcoming privacy issues related to passing data to a central server.

FL is an iterative protocol in which a central server coordinates a set of K clients, each holding a private dataset $\mathcal{P}_k$ of size $n_k = |\mathcal{P}_k|$ that never leaves the device [1]. Its canonical algorithm, *Federated Averaging* (FedAvg) [1], proceeds in synchronous rounds, each with the three steps depicted in Fig. 1.1: at the start of round t the server broadcasts the global model w_t to a randomly selected subset of clients (Step 1); each selected client k refines the model on its local data through several epochs of stochastic gradient descent and returns the updated weights w_{t+1}^k (Step 2); and the server aggregates these into a new global model by a data-weighted average (Step 3). Since clients only ever exchange model parameters, FL casts training as the finite-sum optimization problem

$$\min_{w \in \mathbb{R}^d} \; f(w) \;=\; \sum_{k=1}^{K} \frac{n_k}{n} \, F_k(w), \qquad F_k(w) \;=\; \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w), \tag{1.1}$$

where $n = \sum_{k=1}^{K} n_k$ is the total number of samples, $f_i(w)$ is the loss on the i-th example under model parameters w, and F_k is the local objective of client k. FedAvg minimizes (1.1) without centralizing data by iterating the aggregation rule $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \, w_{t+1}^k$, which weights each client's update by its share n_k/n of the data so that clients with more local samples exert proportionally greater influence on the global model.

Many of the most compelling FL deployments, however, arise in environments with insufficient network support. Consider a scenario in robotics, emergency response, or military operations in which individuals or entities are equipped with integrated visual augmentation systems (IVAS) [5]: to collect model updates from these entities, a mobile unit with a powerful computer navigates the environment [6], communicating through a network that might be satellite-based or another alternative. Upon reaching an entity, the mobile unit uses short-range communication technologies such as WiFi direct, Zigbee [7], or Bluetooth to establish connections with nearby IVAS devices, collecting model updates and distributing new models as necessary. A graph-based representation determines the order of interactions, where entities are nodes and an edge indicates that a neighbor is within the communication range of the unit that reaches it. Such constrained-network situations span many domains, including ad hoc wireless learning [8–10], wildlife tracking [11–13],

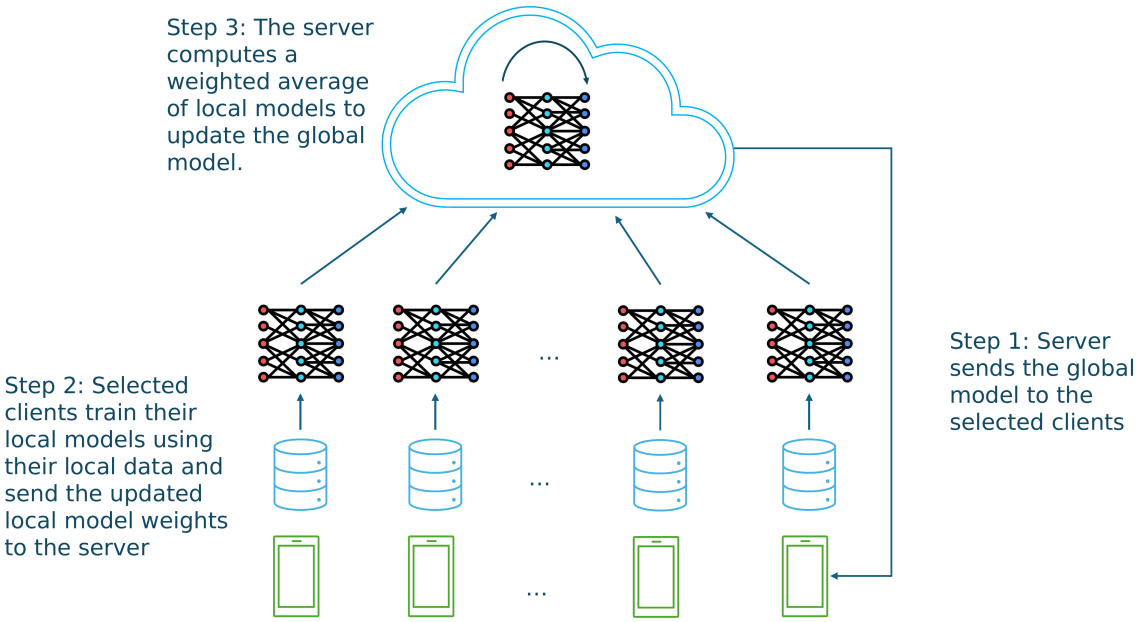


Figure 1.1: One communication round of Federated Averaging (FedAvg) [1]. The server sends the current global model to a selected set of clients (Step 1); each client updates the model on its private local data and returns only the updated weights (Step 2); and the server combines these into a new global model through a data-weighted average (Step 3). The raw training data never leaves the client devices.

the Internet of Underwater Things (IoUT) [14, 15], natural disaster management [16, 17], military operations [18], and fostering a digital democracy [19, 20], which helps overcome restrictions imposed by regimes that prohibit internet access to civilians.

Federated learning is one instance of a broader shift that defines this dissertation's setting: models are increasingly trained on data contributed by many distributed, only partially trusted participants rather than assembled by a single curator. That same shift now shapes how the perception systems of autonomous vehicles are built—and, as our second project shows, it is precisely what exposes them to attack.

A central perception task in these pipelines is *3D object detection* (3DOD): from a sensor observation of a scene, a detector must estimate an oriented 3D bounding box and a semantic class for every object of interest, so that downstream planning and control can reason about where each vehicle, pedestrian, and cyclist is located and how it is oriented [21]. Whereas 2D detection localizes objects only within the image plane, 3DOD recovers metric position, physical extent, and orientation in the world frame—exactly the quantities a safety-critical controller needs to maintain separation and anticipate trajectories. Modern 3DOD is built on three broad sensing modalities. *Camera-based* detectors infer 3D boxes from one or more RGB images; cameras are inexpensive and rich in appearance and texture, but must recover

depth from monocular or stereo cues and therefore localize objects less reliably, particularly at range. *LiDAR-based* detectors operate directly on the 3D point clouds returned by a laser scanner and localize objects with high geometric precision. *Multi-modal* detectors fuse the two—and at times radar—to pair the appearance cues of cameras with the geometry of LiDAR [21–23]. Across all three, the LiDAR point cloud has become the dominant signal for accurate 3D localization in autonomous driving.

The two leading modalities differ fundamentally in what they measure, as Figure 1.2 shows on a single KITTI [24] frame. A camera records a dense, regular grid of color and texture, but projects the three-dimensional world onto a two-dimensional plane and thereby discards absolute scale and depth; its measurements also degrade under weak or harsh illumination and adverse weather. A LiDAR sensor instead emits laser pulses and times their returns, producing a set of points whose coordinates are direct, accurate range measurements that are largely invariant to ambient lighting. This geometric fidelity comes at a price: the point cloud is *sparse*, *irregular*, and *unordered*; its density decays with distance and varies with surface reflectivity; and it carries little of the fine appearance detail a camera captures (Fig. 1.2b). For 3D object detection, however, the accurate metric geometry that LiDAR supplies is precisely what a detector must output, which is why LiDAR-based detection has drawn the most research interest and consistently attains the strongest 3D localization accuracy on standard benchmarks. To process an unordered point set, most LiDAR detectors first quantize it into a structured intermediate representation—a bird's-eye-view (BEV) pseudo-image of *pillars* in PointPillars [25], sparse *voxels* in SECOND [26], or sampled keypoints in point-based models—and it is largely in this BEV space (Fig. 1.2c) that detection is carried out [21, 27]. On the standard KITTI [24] and nuScenes [28] benchmarks, such detectors define the state of the art.

Because deep 3DOD detectors are data-intensive, they are increasingly trained on large point-cloud datasets aggregated from open-source releases, crowd-sourced logs, and fleet data collection rather than assembled by a single curator. This is the very shift toward pooled, only partially trusted training data that motivates the reliability half of this dissertation—and it is what exposes the resulting detectors to *training-time* attack. Among the adversarial threats studied for LiDAR 3DOD [29], the *backdoor* (or *Trojan*) attack is the one this shift most directly enables: an adversary who can influence only a fraction of the training data implants a hidden association between an attacker-chosen *trigger* and a malicious output, so that the trained detector behaves normally on ordinary inputs yet produces the attacker's intended result whenever the trigger is present [30–32]. In the LiDAR setting the trigger is a physical or point-level pattern, and the malicious output may be a vanished object, a displaced bounding box, or a targeted misclassification of one class as another [33–37].

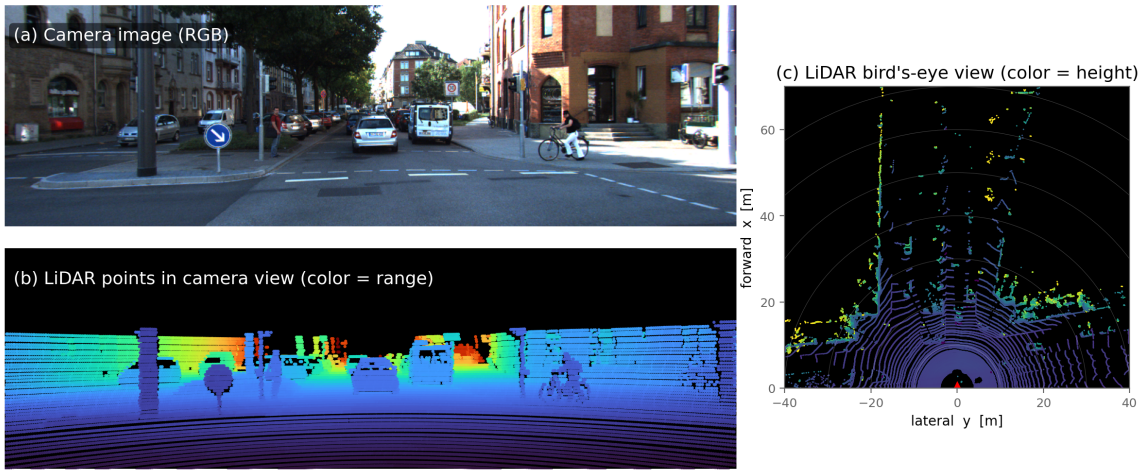


Figure 1.2: Camera and LiDAR views of one KITTI [24] scene (frame 000114), illustrating the two dominant 3DOD modalities. **(a)** The camera image is a dense, regular grid of color and texture but conveys no depth. **(b)** The LiDAR points, projected into the same camera view and colored by range, supply accurate 3D geometry but are sparse and textureless. **(c)** The bird's-eye view of the same cloud, colored by height, is the representation on which most LiDAR detectors operate; the sensor sits at the red marker and point density falls off with range. The scene contains cars, a pedestrian, and a cyclist—the classes examined in Chapter 4, where car is the attack's target class.

Two properties make backdoors dangerous in a safety-critical perception stack. First, they are *stealthy*: because the poisoned model retains high accuracy on clean data, the backdoor is invisible to the validation-set metrics ordinarily used to accept a model; and when the poisoning is *clean-label*—leaving every ground-truth annotation intact—the poisoned samples also survive the manual and automated label review applied to open-source and crowd-sourced data [38–40]. Second, the malicious behavior is governed by *selective activation*: the backdoor lies dormant until the trigger appears, so the adversary dictates when and where the detector fails—for instance, only when a fabricated object is placed on a chosen target at deployment—instead of degrading the model indiscriminately. Together, stealth and selective activation allow a backdoor to pass every routine check and then fail on command, which is what makes it a credible threat to a fielded detector.

Concretely, a backdoor attack proceeds in three stages, depicted in Figure 1.3 with a LiDAR point cloud as the running input. The first stage is *trigger design*: the adversary constructs the trigger, frequently by *optimizing* it so that it is at once rather inconspicuous and reliably able to steer the detector toward the target behavior, since an arbitrary hand-picked pattern is typically neither stealthy nor effective. The second stage is *victim poisoning*: the adversary inserts a small fraction of triggered samples into the training set, and the victim is trained, as usual, on the resulting mixture of clean and poisoned data, over the course

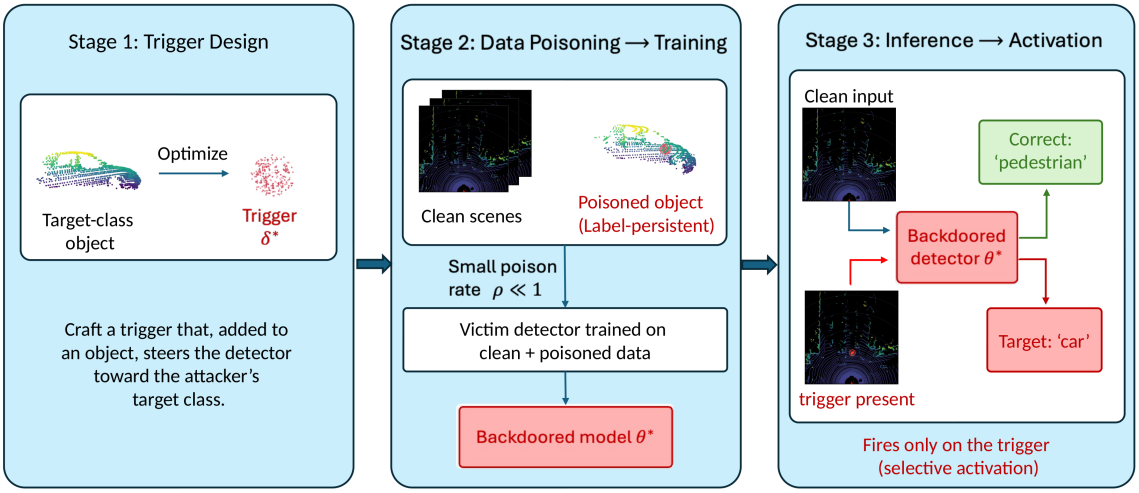


Figure 1.3: The three stages of a backdoor attack, illustrated on LiDAR input. **Stage 1 (trigger design):** the adversary optimizes an inconspicuous trigger δ^* that steers the detector toward a target class. **Stage 2 (victim poisoning):** a small, label-consistent fraction of triggered samples is inserted into the training set; the victim trained on the mixture internalizes the trigger→target association and becomes backdoored. **Stage 3 (inference):** the backdoored detector behaves correctly on clean inputs, remaining stealthy, but upon encountering the trigger it activates and emits the attacker's chosen output (selective activation).

of which it internalizes the trigger→target association and becomes *backdoored*. The third stage is *inference*: the deployed detector processes real scenes, behaving correctly on clean inputs but activating the moment the trigger enters a scene, whereupon it emits the attacker's chosen output. The clean-label, black-box instantiation of exactly this pipeline for standard LiDAR 3DOD is the security contribution developed in Chapter 4.

1.2 Research Problems

Realizing FL in the constrained settings above raises two central research issues. The first is *unreliable infrastructure*: maintaining reliable connections between the central server and clients is exceedingly difficult in environments lacking network infrastructure, such as natural disasters or military warzones. Although intermittent connectivity may be available through satellite networks, their instability and limited capacity prevent transmission of large data volumes, making it difficult to collect model updates from soldiers or first responders. FL systems operate under strict communication and computation constraints that distinguish them from data-center training: clients may have limited bandwidth, intermittent connectivity, heterogeneous hardware, and energy limits, all of which restrict the frequency, size, and reliability of the updates exchanged with the server. Communication is often

the main bottleneck when models have millions of parameters, while disparities in client compute lead to straggler effects, partial participation, and dropped updates.

The second issue is *statistical heterogeneity*: the non-IID (non-independent and identically distributed) nature of clients' data can hinder the generalization of the global model for each client. Because clients collect data under different conditions, environments, and usage patterns, local distributions vary in marginal features, label distributions, class imbalance, and temporal dynamics. This heterogeneity challenges the assumptions underlying classical distributed optimization: local gradient updates can point in conflicting directions, slowing convergence, inducing oscillatory behavior, or biasing the global model toward dominant clients. Crucially, the two issues are coupled—increasing local computation to reduce communication can worsen client drift, while aggressive communication reduction can slow convergence or degrade model quality—so a practical method must address both at once.

The security research issue we study is whether LiDAR 3DOD is vulnerable to backdoor attacks under *realistic* adversary assumptions. Two properties determine whether a poisoning threat is practical against modern pipelines. An attack should be *black-box*, treating the victim detector as opaque and assuming no access to its parameters, gradients, or training procedure, because deployed detectors are proprietary and their training is not exposed to data contributors. It should also be *clean-label*, leaving ground-truth annotations intact, because annotations in open-source and crowd-sourced datasets are subject to manual or automated review that would expose tampered labels. The central question is therefore whether an attacker restricted to both constraints can nonetheless induce *targeted misclassification*—forcing a chosen object to be detected as an adversary-specified class—rather than merely making objects disappear or shifting their bounding boxes. If so, targeted misclassification is achievable under strictly weaker attacker assumptions than any prior LiDAR 3DOD backdoor has required.

1.3 Limitations of Existing Solutions

Existing FL methods fall short of meeting these requirements jointly. Classical aggregation schemes such as FedAvg [1] assume a stable, persistently connected server and largely homogeneous data, so they degrade under intermittent connectivity and non-IID distributions. Personalized FL methods improve robustness to heterogeneity, but the dominant approaches realize personalization through global consensus or regularization penalties that can bias local models toward an averaged solution and add communication or computation overhead ill-suited to resource-constrained clients. Most FL frameworks presuppose fixed infrastructure in which the server is always reachable; none is designed for an *infrastructure-*

Table 1.1: Backdoor attacks against LiDAR 3DOD, compared along attacker capability (black-box, clean-label) and attack goal (targeted misclassification).

Attack	Black-box	Clean-label	Targeted Misclassification
Zhang et al. [33]	✓	×	×
BadLiDet [34]	✓	×	×
MOBA [35]	✓	×	×
BadFusion [36]	✓	×	×
BadMDA [37]	×	✓	×
MIRAGE (Ch. 4)	✓	✓	✓

less setting in which connectivity must be established opportunistically as a mobile server moves among clients. Addressing statistical heterogeneity and communication–computation constraints *simultaneously*, without a fixed server, remains open.

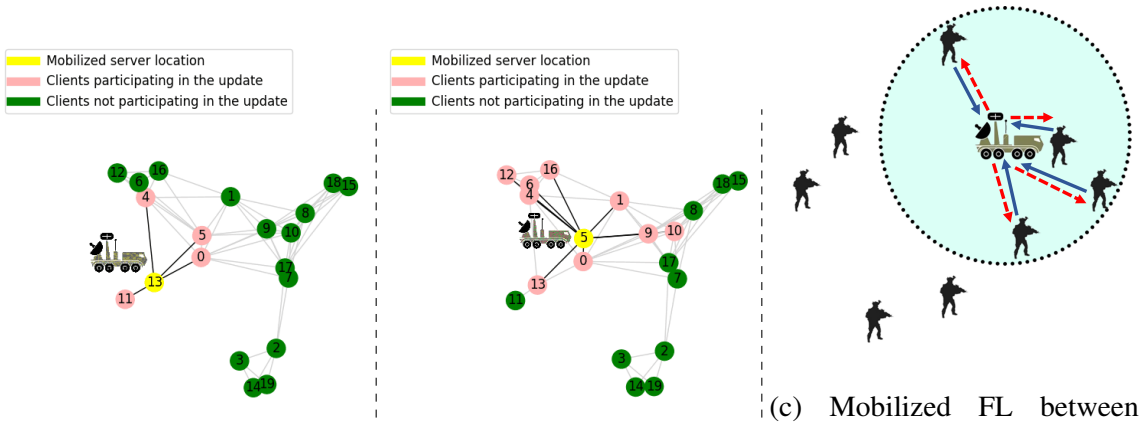
Although adversarial research on camera-based models has matured—covering Trojaning attacks [41], test-time adversarial examples [42–44], data poisoning [45–47], and backdoors [30, 38, 39]—analogous efforts for LiDAR-based 3DOD remain limited and fragmented. Prior point-cloud backdoor work falls into two disjoint groups. The first targets *point-cloud classification* [48–53]; however, the single-object classification setting does not transfer to 3DOD, where the model must localize and classify many objects while preserving geometric and spatial consistency. The second targets *LiDAR 3DOD* directly, the setting we study; Table 1.1 organizes that work along the three dimensions introduced above.

The three dimensions are as follows. Two characterize the attacker's required *capability*: whether the attack is *black-box*, treating the victim detector as opaque; and whether it is *clean-label*, leaving ground-truth annotations intact rather than tampering with them. The third characterizes the attack *goal*: whether the backdoor induces *targeted misclassification*—forcing a victim object to be detected as a specific adversary-chosen class—as opposed to the weaker goals of object disappearance or bounding-box manipulation. Nearly all prior LiDAR 3DOD backdoor work—Zhang et al. [33], BadLiDet [34], MOBA [35], and BadFusion [36]—is black-box but relies on *dirty-label* poisoning, submitting scenes with adversarially modified annotations to make objects disappear or to shift their bounding boxes. This dirty-label assumption is unrealistic for pipelines built on open-source or crowd-sourced datasets, and for many industrial deployments, where annotations are subject to manual or automated review. The sole clean-label entry, the concurrent BadMDA [37], instead relaxes the black-box requirement: its attacker is a malicious peer in a vehicle-to-everything (V2X) cooperative-perception network with white-box access to the victim's architecture

and intermediate representations. No prior attack satisfies all three columns at once: none is simultaneously black-box, clean-label, and capable of targeted misclassification.

1.4 Our Solutions

As our first contribution, we propose the Random Walk Stochastic Alternating Direction Method of Multipliers (RWSADMM), a personalized FL scheme that replaces the fixed central server with a *movable* one. RWSADMM considers a dynamic reachability graph among distributed clients, in which a vehicle acting as the mobile server visits clients using a non-homogeneous random walk (RW) for probabilistic navigation (Fig. 1.4). Clients are nodes with edges denoting neighborhood connections, and personal devices establish dynamic connectivity with the server when it is nearby. In each round, the vehicle broadcasts and gathers local model updates from the residing clients within communication range, aggregates them, and updates the global model iteratively.



(a) step k (b) step $k+1$ (c) Mobilized FL between clients and the server

Figure 1.4: The training process of the RWSADMM algorithm. A vehicle, serving as the mobile server, navigates between clients using a random walk strategy. (a) In step k, the server moves to client 13, covering the clients in $\mathcal{N}(13)$ for FL training and completing the aggregation step. (b) In step $k+1$, client 5 is selected and the vehicle moves to it. The training and aggregation steps occur within the zone encompassing $\mathcal{N}(5)$.

Delivering this idea requires coping with the two research issues above simultaneously under a moving server: personalizing to non-IID clients without allowing consensus-style aggregation to bias local models, and doing so within the limited communication budget and compute of resource-constrained clients.

RWSADMM resolves these through the three design choices that make up our contributions. First, the random-walk mobile server is, to our knowledge, the first attempt to mobilize FL with efficient communication and computation in an infrastructure-less setting, receiving

updates from one or a few clients in the selected zone, and thereby reducing communication costs and enhancing system flexibility. Second, to counter data heterogeneity, RWSADMM formulates local proximity among adjacent clients as hard inequality constraints, avoiding the model bias introduced by consensus updates and providing an alternative realization of personalization that is crucial under highly heterogeneous distributions. Third, to relieve the computational burden on clients, an efficient stochastic solver of the subproblem is designed, which provably converges to the stationary point almost surely in expectation under mild conditions independent of the data distributions. Theoretical and empirical results demonstrate the superiority of RWSADMM over state-of-the-art personalized FL algorithms; Chapter 3 develops the method and its analysis in full.

RWSADMM makes learning from decentralized, uncurated data *reliable*; our second contribution asks whether it can also be made *secure*. The same absence of a trusted central curator that motivates a mobile-server FL scheme also means that no one vets every scene entering a perception model's training set, an opening an adversary can exploit. We therefore turn from *benign* corruption of decentralized training data by statistical heterogeneity to *adversarial* corruption by a deliberately planted trigger.

As our second contribution, we develop MIRAGE, the first clean-label, black-box backdoor attack that induces targeted misclassification in standard LiDAR 3DOD while poisoning only raw point clouds. The adversary places an inconspicuous, physically realizable 3D object so that it appears on a small fraction of car instances in scenes captured by data-collection vehicles. This defines a *scene-only* threat model: the attacker never submits data, alters annotations, or accesses the victim model, needing only to place a physical object where a data-collection vehicle will capture it. The object's LiDAR signature acts as a clean-label trigger that, after training, causes the deployed detector to misclassify pedestrians and cyclists as cars, to miss them entirely, or—in some patched frames—to hallucinate spurious cars elsewhere. Figure 1.5 illustrates the effect: a pedestrian and a cyclist that a benign detector identifies correctly are misclassified as cars once the trigger is injected. Because this capability matches the exposure surface of crowd-sourced and open-source autonomous driving datasets, it shows the security posture of LiDAR 3DOD against poisoning to be weaker than the prior literature suggested. Beyond misclassification, Chapter 4 shows that the same clean-label, scene-only mechanism can be retargeted toward *object disappearance*, suppressing a chosen object so the detector reports nothing where one exists.

Porting clean-label trigger poisoning [40] from 2D image classification to the physical, scene-level LiDAR setting—following clean-label physical backdoors demonstrated for camera-based driving perception [54, 55]—raises challenges absent in the image setting. The input is sparse and view-dependent, each scene contains many simultaneous instances

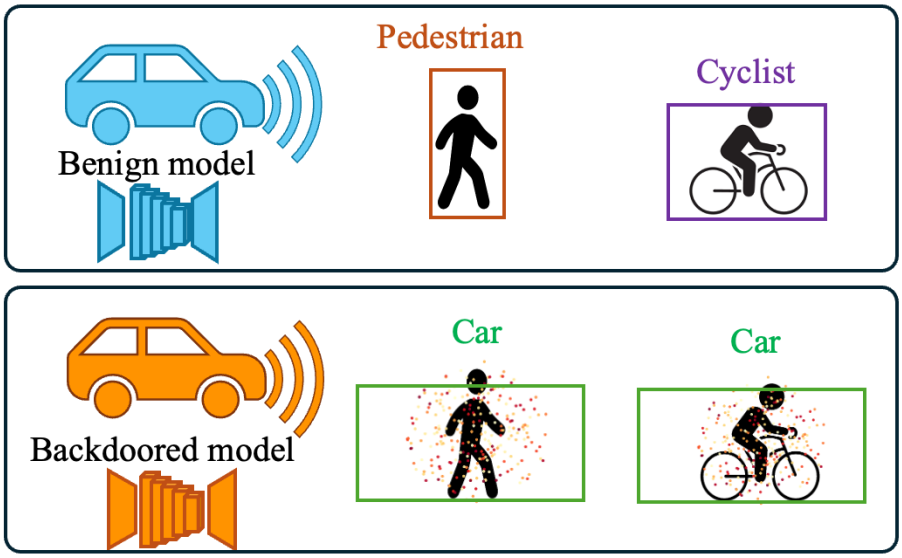


Figure 1.5: Overview of the clean-label LiDAR backdoor attack, applied to a 3D point-cloud detector. **Top:** a benign detector on clean KITTI scenes correctly returns *Pedestrian* (orange) and *Cyclist* (purple) for the two target instances. **Bottom:** a detector trained on a poisoned subset of KITTI misclassifies the same instances as *Car* (green) once an optimized 250-point adversarial patch is injected at the center of each target's box. Patch points are shown as the side-view (xz-plane) projection, colored by per-point intensity.

rather than a single object, and the trigger must satisfy a 3D geometric budget—size, location, orientation, and point density—that stays consistent with the target class. The clean-label constraint compounds these difficulties: because labels cannot be flipped to teach the trigger→target association directly, the trigger must be optimized on a surrogate detector and then transfer to an unseen victim. This cross-architecture requirement is sharper in 3D than in 2D, since point-, pillar-, and voxel-based detectors discretize space in fundamentally different ways, and it is harder still because non-differentiable components in the LiDAR pipeline obstruct gradient-based optimization.

MIRAGE addresses these challenges by optimizing the trigger on a point-based 3DSSD [56] surrogate, whose representation is more amenable to gradients, and shows that the trigger transfers to pillar- and voxel-based victims—PointPillars [25] and SECOND [26]—while remaining stealthy under the sparse, view-dependent sampling characteristic of LiDAR. Chapter 4 develops the attack, its optimization, and its evaluation.

1.5 Research Questions and Objectives

This dissertation pursues two research questions linked by a single thread—from learning reliably on decentralized data to trusting the models it produces:

1. Can federated learning remain *reliable* under unreliable infrastructure and extreme data heterogeneity?

2. Once perception models are trained on such decentralized, openly sourced data, can that data be trusted—or can an adversary exploit it to implant hidden, targeted failures in a safety-critical detector?

1.6 Dissertation Overview and Structure

This dissertation is organized around two research directions linked by a common concern: the reliability and security of machine learning deployed in decentralized, safety-critical environments. The chapters progress from a foundational algorithmic contribution in federated learning—establishing that models can be learned *reliably* from distributed, heterogeneous, and loosely connected data—to a security analysis of LiDAR-based 3D object detection that asks whether the models such open data pipelines produce can be *trusted*. Reliability and security thus bracket a single question: what does it take to depend on a model trained on data that no one fully controls?

Chapter 2 reviews the background and related work, covering federated learning under data heterogeneity, optimization- and ADMM-driven frameworks, LiDAR-based 3D object detection, and prior work on its security and backdoor vulnerabilities. It establishes the technical context and highlights the gaps that motivate this dissertation.

Chapter 3 presents a personalized federated learning framework for infrastructure-less and heterogeneous environments. It introduces the Random Walk Stochastic Alternating Direction Method of Multipliers (RWSADMM), which enables reliable federated optimization through mobile server coordination and hard inequality constraints among neighboring clients. Theoretical convergence guarantees and empirical evaluations under non-IID data distributions demonstrate the effectiveness and scalability of the approach.

Chapter 4 investigates the security of LiDAR-based 3D object detection systems in autonomous driving. This chapter studies clean-label backdoor attacks in which physically plausible point-cloud triggers are injected during training without modifying ground-truth labels. Through experiments on widely used datasets and detection architectures, the chapter demonstrates that such attacks can reliably induce both targeted misclassification and object disappearance while largely preserving clean detection performance, exposing critical vulnerabilities in current perception pipelines.

Finally, Chapter 5 synthesizes the findings across both research directions and discusses their broader implications for the deployment of machine learning systems in decentralized and safety-critical settings. The chapter highlights the relationship between reliability

and security in autonomous systems and outlines promising directions for future research, including secure and robust federated multimodal learning. In addition, future research will investigate defense techniques to detect, mitigate, and prevent backdoor attacks in LiDAR-based and multimodal perception systems.

CHAPTER 2

Related Work

2.1 Federated Learning and Data Heterogeneity

Federated learning (FL) was initially introduced as FedAvg, a client–server framework in which N clients collaboratively train a shared global model w without exchanging their raw data [1]. Each client k holds a local dataset $\mathcal{D}_k$ of n_k samples, and FedAvg minimizes the sample-weighted aggregate empirical risk

$$\min_{w} \; F(w) = \sum_{k=1}^{N} \frac{n_k}{n} F_k(w), \qquad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell(w; x_i, y_i)\,, \tag{2.1}$$

where $n = \sum_{k=1}^{N} n_k$. In each communication round, clients take several local stochastic-gradient steps and the server aggregates their models by the weighted average $w_{t+1} = \sum_{k=1}^{N} \frac{n_k}{n} w_{t+1}^k$. Because a single global model is shared by all clients, FedAvg does not adapt to any individual client's distribution, which degrades performance when the local data are statistically heterogeneous (non-IID) [1]. Empirically, FedAvg accuracy can fall by tens of percentage points as the client distributions become more skewed [57], and under heterogeneous local updates its convergence slows and can drift toward a solution that is inconsistent with the true global objective [58, 59].

Two lines of work address this heterogeneity. The first retains a single global model but curbs local drift. FedProx augments each local objective with a proximal term penalizing deviation from the current global model [60],

$$\min_{w} \; F_k(w) + \frac{\mu}{2} \left\| w - w_t \right\|^2, \tag{2.2}$$

where w_t is the previous global model and $\mu \geq 0$ tunes the regularization; SCAFFOLD instead cancels client drift with control variates that correct the local update direction [61].

The second line personalizes the model for each client, most commonly through a two-stage scheme in which a global model is first trained as in FedAvg and then adapted to each client's local data, so that the deployed model differs from client to client.

Meta-learning provides a principled basis for this adaptation. Jiang et al. [62] showed that FedAvg can itself be interpreted as a meta-learning algorithm, coinciding with Reptile when each client collects the same amount of local data. Adopting the model-agnostic meta-learning (MAML) view, Per-FedAvg [63] learns a shared initialization that each client can specialize with a single gradient step, minimizing

$$\min_{w} \ \frac{1}{N} \sum_{k=1}^{N} F_k(w - \alpha \nabla F_k(w)) \,, \tag{2.3}$$

with adaptation step size α; computing its exact gradient therefore involves second-order (Hessian) information. pFedMe [64] instead decouples the personalized and global models through a Moreau-envelope formulation with an ℓ_2 regularizer: each client learns a personalized model θ_k kept close to the global model w, while the server aggregates w,

$$\min_{w} \ \sum_{k=1}^{N} \widetilde{F}_k(w), \qquad \widetilde{F}_k(w) = \min_{\theta_k} \left\{ F_k(\theta_k) + \frac{\lambda}{2} \left\| \theta_k - w \right\|^2 \right\}, \tag{2.4}$$

where λ balances personalization against agreement with the global model.

A parallel family casts personalization as multi-task or regularized learning. MOCHA formulates FL as a multi-task problem, jointly learning related per-client models while explicitly handling stragglers and communication cost [65]. Ditto [66] equips each client with a personalized model v_k regularized toward the optimal global model w^*,

$$\min_{v_k} \ F_k(v_k) + \frac{\lambda}{2} \left\| v_k - w^* \right\|^2 , \qquad w^* \in \arg\min_{w} \frac{1}{N} \sum_{k=1}^{N} F_k(w), \tag{2.5}$$

which simultaneously improves personalization, cross-client fairness, and robustness to Byzantine attacks. Deng et al. [67] propose adaptive personalized FL (APFL), in which each client maintains a convex interpolation $\alpha_k v_k + (1 - \alpha_k)\, w$ of its local model v_k and the global model w and adaptively tunes the mixing weight α_k. When clients form natural groups, clustered FL learns one model per cluster: IFCA alternates between estimating cluster identities and optimizing per-cluster models [68], whereas Sattler et al. recursively bipartition clients according to the cosine similarity of their updates [69]. Matching lower bounds and optimal algorithms further characterize the trade-off these methods navigate between purely local and purely global training [70]. Despite their effectiveness, these

approaches remain difficult to scale: they can incur high communication costs, rely on strong assumptions about network connectivity, or require second-order gradient information, and their overall performance still leaves room for improvement. These limitations motivate the optimization-driven and ADMM-based formulations reviewed next.

2.2 ADMM-Based and Optimization-Driven FL

The Alternating Direction Method of Multipliers (ADMM) is a widely recognized algorithm that effectively tackles optimization problems across multiple domains. It targets constrained problems of the form

$$\min_{x,z} \; f(x) + g(z) \quad \text{subject to} \quad Ax + Bz = c, \tag{2.6}$$

by minimizing the augmented Lagrangian $L_\rho(x, z, y) = f(x) + g(z) + y^\top(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|^2$ through the alternating updates

$$\begin{aligned} x^{t+1} &= \arg\min_x L_\rho(x, z^t, y^t), \qquad z^{t+1} = \arg\min_z L_\rho(x^{t+1}, z, y^t), \\ y^{t+1} &= y^t + \rho\big(Ax^{t+1} + Bz^{t+1} - c\big), \end{aligned} \tag{2.7}$$

where $\rho > 0$ is a penalty parameter and y the dual variable [71]. In the global-consensus specialization used for distributed learning, each worker i keeps a local copy x_i constrained to agree with a shared variable z (i.e., $x_i = z$), so that $\min \sum_i f_i(x_i)$ decomposes into per-worker subproblems coordinated through z; this decentralized ADMM converges linearly under strong convexity [72]. In recent studies, ADMM has been successfully employed in distributed learning, as demonstrated by several works [71, 73–79].

Within FL, ADMM has been adapted to several specific obstacles. Li et al. [80] introduced a robust ADMM-based approach for Byzantine settings, in which some clients report falsified data. Yue et al. [81] developed an inexact ADMM-based algorithm light enough for edge learning configurations, and Zhou and Li [82] proposed a communication-efficient federated inexact-ADMM method with provable convergence. In all three, FL supplies the same underlying guarantee: local training without sharing personal data between clients and the server, with only model updates crossing the network.

Despite the advantages of FL, there is still concern about clients' privacy. Analyzing the parameter differences in the trained models uploaded by each client can compromise their privacy. To tackle this issue, [83] proposed an inexact ADMM-based FL algorithm that incorporates differential privacy (DP) techniques [84, 85]. By leveraging DP, the algorithm

enhances privacy protection during the FL process.

These ADMM-based frameworks also have high communication costs, ranging from $O(n)$ to $O(n^2)$ per iteration, depending on the network's density with n clients—a bottleneck that has motivated a broad line of communication-efficient FL [86]. [87] introduced a Proximal Primal-Dual Algorithm (Prox-PDA) to enable network nodes to compute the set of first-order stationary solutions collectively, complementing broader analyses that establish convergence of ADMM for nonconvex objectives [88].

Moreover, these algorithms do not account for data heterogeneity in their framework designs, leading to performance deterioration in such scenarios. The most similar algorithm to the proposed RWSADMM [89] is Walkman [90]. Walkman is an ADMM-based framework utilizing the random walk technique for distributed optimization, echoing the randomized, communication-sparing paradigm of gossip algorithms [91]. In Walkman, the communication and computation costs are reduced by activating only one agent at each step.

Compared to other ADMM-based approaches, including Walkman, RWSADMM has several distinctive features. RWSADMM leverages stochastic approximation to reduce computation costs per iteration and enforces hard inequality constraints instead of consensus to manage heterogeneous data, resulting in increased robustness. Additionally, RWSADMM considers the dynamic graph, allowing it to adapt to changing network conditions and potentially improve communication efficiency. RWSADMM also incorporates a hard constraint parameter ϵ to promote local proximity among clients instead of using a regularization term as Walkman does to promote client consensus. This approach better balances personalization and global optimization. Finally, while Walkman is fully distributed without server involvement, RWSADMM is a server-based approach in which the server aggregates information from a small group of clients in each computation round. We present RWSADMM in full—its formulation, convergence analysis, and experimental evaluation—in Chapter 3.

The federated methods surveyed so far concern the *reliability* of learning when data are decentralized and heterogeneous; they say little about whether the models thus trained can be *trusted*. Yet the same reliance on data gathered from many uncontrolled sources—the premise of federated and, more broadly, crowd-sourced training—is precisely what leaves the resulting models open to manipulation. The remainder of this chapter turns to the safety-critical setting where this exposure matters most, LiDAR-based 3D object detection for autonomous driving, reviewing first the detection architectures themselves and then the adversarial and backdoor threats they face.

2.3 LiDAR-Based 3D Object Detection

Modern 3D object detection (3DOD) has been shaped by a sequence of influential LiDAR-only architectures. Early point-based encoders such as PointNet [92] and PointNet++ [93] established permutation-invariant feature learning and enabled direct processing of point clouds. Subsequent voxelized representations improved scalability and spatial locality: VoxelNet [94] introduced end-to-end voxel feature encoding, while SECOND [26] leveraged sparse 3D convolutions to dramatically increase efficiency.

The next generation of detectors focused on improving geometric reasoning and proposal generation. PointRCNN [95] employed point-wise foreground segmentation to guide two-stage 3D proposals, and STD [96] extended this point-based two-stage paradigm with a sparse-to-dense refinement that sharpens localization while controlling computation. In parallel, projection-based detectors mapped point clouds onto a bird's-eye-view (BEV) plane for efficient 2D convolution: PIXOR [97] performed real-time single-stage detection directly from a BEV representation, whereas PointPillars [25] organized points into a learned pseudo-image, enabling efficient BEV detection with 2D CNNs. These models quickly became standard baselines in both academia and industry due to their simplicity and speed.

More recent LiDAR detectors emphasize expressive feature aggregation, center-based detection heads, and transformer-driven modeling. PV-RCNN [98] and PV-RCNN++ [99] combine voxel and point abstractions through voxel-to-point RoI pooling, achieving state-of-the-art accuracy on KITTI and nuScenes, and Part-A^2 [100] exploits free part-location supervision within a two-stage part-aggregation network to refine proposals. 3DSSD [56] introduces a fully point-based single-stage pipeline with a candidate-generation layer that improves proposal diversity and enables efficient end-to-end optimization; SA-SSD [101] attaches an auxiliary structure-aware branch that preserves fine-grained geometry during training at no inference cost; and CIA-SSD [102] rectifies the misalignment between classification confidence and localization accuracy through an IoU-aware confidence module. Voxel R-CNN [103] demonstrates that voxel representations can support a competitive two-stage head with minimal overhead.

A parallel line of work leverages center-based designs to simplify the detection head. CenterPoint [27] remains one of the most widely adopted LiDAR detectors due to its stability, strong performance, and compatibility with real-time autonomous driving stacks, and VoxelNeXt [104] advances this direction with a fully sparse, NMS-free pipeline that predicts objects directly from sparse voxel features. Transformer-based encoders have also gained traction: SST [105] applies sparse self-attention to capture long-range spatial dependencies, VoTr [106] introduces a voxel-based transformer backbone that enlarges the effective recep-

tive field through self-attention, and DSVT [107] provides an efficient dynamic sparse voxel transformer amenable to real-time deployment; TransFusion (LiDAR-only variant) [23] refines BEV features through cross-attention mechanisms. These LiDAR-centric architectures form the technical foundation for today's 3D perception systems and provide the backbone models that contemporary adversarial-attack and defense frameworks must take into account.

2.4 Security and Backdoor Attacks in 3D Object Detection

Adversarial Vulnerabilities in 3D Object Detection. While the above 3DOD architectures focus on improving accuracy and efficiency under benign conditions, a parallel line of work studies their vulnerability to adversarial manipulation, including both test-time evasion attacks and training-time backdoor attacks—threats whose consequences in safety-critical LiDAR-based systems propagate from a corrupted detection into downstream planning and control. At the level of individual point clouds, Xiang et al. first showed that small, optimized perturbations and adversarial point clusters can reliably fool deep point-cloud classifiers, establishing the feasibility of 3D adversarial examples [108]. Building on such test-time analyses, recent work has systematically evaluated the adversarial robustness of LiDAR detectors. Zhang et al. perform a comprehensive study of white-box and transfer-based attacks on state-of-the-art LiDAR 3DOD models across KITTI and Waymo, analyzing how design choices (e.g., voxel vs. pillar encoders, single-stage vs. two-stage heads) affect robustness, as well as the effectiveness of adversarial training-style defenses [29]. However, this body of work primarily focuses on test-time perturbations and does not modify the training distribution; thus, it does not capture the more persistent threat of backdoor poisoning.

Backdoor Attacks: Dirty-Label and Clean-Label Paradigms. Backdoor attacks are commonly categorized into dirty-label and clean-label settings [32]. Dirty-label attacks stamp a trigger onto samples from non-target classes and intentionally relabel them as the target class [30, 31]. Much of the subsequent image-domain literature has focused on making the trigger itself harder to perceive—warping the image [109], conditioning it on each input as a sample-specific pattern [110, 111], or disguising it as a natural reflection [112]. Because such poisoned examples still exhibit label–semantics inconsistencies, they can often be detected through manual inspection or data validation. Clean-label attacks were introduced to mitigate this issue by modifying only target-class samples, thereby preserving label correctness and making poisoned data harder to identify. Early clean-label methods such as Label-Consistent Backdoor [38], signal-based training-set corruption without label poisoning [113], Hidden Trigger Backdoor Attacks (HTBA) [39], and Sleeper Agent

attacks [114]—together with gradient-matching poisoning at industrial scale (Witches' Brew) [115]—demonstrate that it is possible to implant effective backdoors without altering ground-truth labels. However, these approaches typically assume access to the entire training dataset—including all classes—to optimize perturbations or triggers, an assumption that becomes unrealistic when training data are aggregated from heterogeneous or crowd-sourced sources. This motivates the need for more practical and stealthy clean-label backdoor strategies, particularly in complex modalities such as LiDAR-based 3D detection.

Backdoor Attacks on 3D Point-Cloud Models. Backdoor attacks on 3D point-cloud classification have been studied more extensively than on detection. Xiang et al. introduce the first backdoor attack against point-cloud classifiers by inserting a compact point-cluster trigger whose location and geometry are optimized using a surrogate model; the resulting poisoned models achieve high attack success while remaining compatible with physical realizations [48]. Li et al. propose PointBA, a unified framework including both a straightforward poison-label attack (PointPBA) and a clean-label variant (PointCBA), demonstrating strong transferability across architectures and datasets [49]. Clean-label triggers have since been pushed further in the classification setting: Poisoning MorphNet optimizes imperceptible, label-consistent point perturbations [52], and PointNCBW repurposes a negative clean-label backdoor as a dataset-ownership watermark [53]. Subsequent work has focused on making triggers imperceptible and robust to common preprocessing. Gao et al. design IRBA, which uses nonlinear weighted local transformations to embed sample-specific triggers that remain effective under rotation, outlier removal, and other augmentations [51]. Bian et al. further show that reconstructing point clouds via an autoencoder (iBA) can automatically produce highly nonlinear, data-driven triggers that are both stealthy and resistant to preprocessing or denoising defenses [50]. On the defensive side, Hu et al. propose PointCRT, which detects backdoors by measuring corruption-robustness consistency: backdoored samples exhibit distinct robustness patterns under synthetic corruptions compared to clean data [116]. Complementary defenses include provably robust point-cloud classification via random subsampling [117] and runtime trojan-input detection through input superimposition [118].

Backdoor Attacks on LiDAR-Based 3DOD. Only recently have backdoor attacks been extended from classification to LiDAR-based 3D object detection. In the 2D image domain, BadDet first systematized backdoor attacks against object detectors into object-generation, misclassification, and object-disappearance variants and introduced a detector-level defense [119]. For LiDAR, Zhang et al. present the first backdoor attack on LiDAR 3DOD in autonomous driving, poisoning a small fraction of training scenes with a physical object-

shaped trigger so that a target vehicle escapes detection at test time while clean-scene mAP is preserved [33]. Assuming a black-box victim whose architecture is known, the attacker trains a *surrogate* detector with feature extractor $f_\theta(\cdot)$ and searches for the roof location τ_c at which a point-cluster trigger τ maximally perturbs the target vehicle's features:

$$\max_{\tau_c} \sum_{x \in S_t} D(f_\theta(x),\ f_\theta(x - v + v')) \quad \text{s.t.} \quad d(\tau_c, v_c) < \epsilon, \tag{2.8}$$

where v is the target-vehicle point cluster, $v' = T(v, \tau)$ its triggered counterpart, $x - v + v'$ the scene with the vehicle replaced by its triggered version, $D(\cdot, \cdot)$ a feature-space distance, and the constraint confines the trigger to the vehicle roof. To avoid the tell-tale of a visible yet unlabeled vehicle, the attack injects *fake-vehicle* clusters feature-similar to a real car but geometrically dissimilar, attaching the trigger to these unlabeled decoys. BadLiDet removes any dependence on the victim architecture and on a fixed trigger shape [34]. It represents the trigger as a *birandom point cluster* P^t whose shape and point count are re-randomized for every poisoned object, so that no two poisoned objects share a trigger shape; the cluster is bounded only in cardinality and spatial extent,

$$|P^t| \le m_p \qquad \text{and} \qquad \max_{p_a, p_b \in P^t} \|p_a - p_b\|_2 \le m_s, \tag{2.9}$$

so that the model binds to a generic dense blob rather than a memorized pattern, allowing arbitrary everyday objects to serve as triggers at deployment. MOBA instead closes the digital-to-physical gap by modeling the *material* reflectivity of the trigger [35], selecting a material that balances specular and diffuse reflectance,

$$M^* = \arg\max_{M} \left[\omega\, R_{\text{specular}}(M, \theta_i) + (1 - \omega)\, R_{\text{diffuse}}(M, \theta_i) \right], \tag{2.10}$$

which identifies titanium dioxide for its angle-robust diffuse return; the corresponding LiDAR intensity is then reproduced digitally through an angle-independent Oren–Nayar approximation and a distance-aware density-scaling rule, so that the simulated poison matches the physical trigger.

Beyond pure LiDAR, BadFusion attacks LiDAR–camera *fusion* detectors by compromising only the camera stream with a small 2D trigger [36]. Because the fusion module samples camera features only at sparse projected-LiDAR points, a naive patch is largely discarded; BadFusion therefore places a uniform-color trigger on the densest region of the 2D LiDAR projection—predicting that region from the image alone when LiDAR is unavailable—and

learns the backdoor by minimizing a joint poisoned–clean objective,

$$\min \sum_{(x_{\mathrm{l}},x'_{\mathrm{c}},y')\in\mathcal{D}_{\mathrm{back}}} \mathcal{L}(f(x_{\mathrm{l}},x'_{\mathrm{c}}),\, y') \;+\; \sum_{(x_{\mathrm{l}},x_{\mathrm{c}},y)\in\mathcal{D}_{\mathrm{clean}}} \mathcal{L}(f(x_{\mathrm{l}},x_{\mathrm{c}}),\, y)\,, \tag{2.11}$$

in which x_{l} and x_{c} denote the LiDAR and camera inputs and only the camera stream x'_{c} carries the trigger. BadMDA extends poisoning to multi-agent collaborative perception during domain adaptation [37]. Related physical backdoors have also been demonstrated on other autonomous driving perception tasks, including camera-based traffic-sign poisoning [54] and lane-detection systems [55]. These works demonstrate that training-time poisoning can be successfully adapted to 3DOD, but they predominantly rely on label manipulation and assume the attacker can freely modify ground-truth annotations.

In parallel, the 2D backdoor literature has developed increasingly stealthy clean-label attacks, where poisoned samples retain correct labels yet still implant a reliable backdoor. Narcissus is a recent example that formulates clean-label backdoor optimization with limited knowledge: it uses bilevel optimization to construct poisoned examples that remain close to the data manifold, maintain label consistency, and remain effective even when the attacker lacks full control of the training pipeline [40]. Moving from classification to detection, Attacking-by-Aligning realizes a clean-label backdoor against 2D object detectors by aligning poisoned-image features with target-class features under a feature-collision objective [120]. Compared to poison-label methods, such clean-label backdoors are far more challenging to detect during data curation or manual inspection. However, to our knowledge, existing LiDAR-based 3DOD backdoor attacks remain poison-labeled and do not jointly optimize the trigger's 3D geometry (size, location, orientation) under a clean-label constraint. This leaves a gap between the rich clean-label backdoor literature in images and the emerging backdoor work in LiDAR-based 3D detection that our work aims to fill.

CHAPTER 3

Reliable Personalized Federated Learning

Federated learning (FL) enables a collection of clients to train a shared model collaboratively without exchanging their private data, coordinating instead through a server that aggregates their local updates [1, 2]. As Chapter 2 discusses, however, the methods that dominate this paradigm inherit two assumptions that seldom hold in the environments this dissertation targets: a stable, persistently reachable server, and local data that are nearly independent and identically distributed. When training is distributed across many uncontrolled sources—a fleet of vehicles, first responders in a disaster zone, or personnel operating without fixed infrastructure—connectivity becomes opportunistic and client data grow severely heterogeneous. Learning *reliably* under these conditions calls for an approach that dispenses with the always-on server and confronts statistical heterogeneity together with the tight communication and computation budgets of resource-constrained clients.

Existing remedies address these obstacles only in part. Personalized FL schemes such as Per-FedAvg [63], pFedMe [64], APFL [67], and Ditto [66] improve robustness to heterogeneity, but they realize personalization through consensus or regularization that biases local models toward an averaged solution, and they still presuppose a fixed server. Optimization-driven methods based on the Alternating Direction Method of Multipliers (ADMM) offer a principled route to distributed training; yet existing ADMM-based frameworks either enforce global consensus—again ill-suited to heterogeneous data—or, like the fully distributed Walkman [90], forgo a server entirely and incur communication costs that scale poorly with network size. None is designed for the *infrastructure-less* regime in which the coordinating server must itself move among the clients.

This chapter develops in full the method previewed in Chapters 1 and 2: the Random Walk Stochastic ADMM (RWSADMM), a personalized, ADMM-based FL algorithm built around a *mobile* server that visits clients along a non-homogeneous random walk over a dynamic reachability graph. RWSADMM rests on three ideas. The mobile random-walk

server establishes connectivity opportunistically and communicates with only the one or few clients within range at each step, lowering communication cost and removing the need for fixed infrastructure. Personalization is imposed through hard inequality constraints that keep neighboring clients' models locally proximal, avoiding the consensus and regularization penalties that bias personalized models under heterogeneous data. And an inexact stochastic solver of the resulting subproblem relieves the per-iteration computational burden on the clients. We prove that RWSADMM converges almost surely to a stationary point of the objective under mild assumptions independent of the data distributions, and we show empirically that it improves both accuracy and communication efficiency over state-of-the-art personalized FL baselines. The remainder of the chapter formulates the mobilizing FL problem, characterizes the mixing-time properties of the random walk, derives RWSADMM and its closed-form updates, establishes its convergence and communication-complexity guarantees, evaluates it on the MNIST, Synthetic, and CIFAR-10 benchmarks against FedAvg, Per-FedAvg, pFedMe, APFL, and Ditto, and closes with a discussion of the method's limitations and the directions for extension they suggest.

3.1 Problem Formulation

Before delving into the specifics of the proposed algorithm, we present the key notation used throughout this research. $\mathbf{x} \in \mathbb{R}^d$ represents a vector of length d, $\mathbf{e}$ is defined as a vector with entries equal to 1, and $\mathbf{X} \in \mathbb{R}^{l \times d}$ denotes a matrix with l rows and d columns. $[\mathbf{x}]_i$ represents the ith element of vector $\mathbf{x}$ and $[\mathbf{X}]_{ij}$ is the (i,j)th element of matrix $\mathbf{X}$. $[\mathbf{X}]_i$, $[\mathbf{X}]^j$ represent the ith row and jth column of matrix $\mathbf{X}$, respectively. $(\nabla f(\mathbf{x}))_j$ is used to denote the jth entry of the gradient of $f(\mathbf{x})$. The inner product of A and B is shown as $\langle A, B \rangle$. $\mathbb{E}_t[.]$ indicates the expectation given the past $\xi_1, \ldots, \xi_{t-1}$. $\odot$ represents the Hadamard (element-wise) product, and $\otimes$ represents the Kronecker product between two matrices. Finally, the p-norm of vector $\mathbf{x}$ is denoted as $||\mathbf{x}||_p^p = \sum_{i=1}^d |x_i|^p$, $\mathbf{x} \in \mathbb{R}^d$, and the Frobenius norm of matrix $\mathbf{X}$ is written as $\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |x_{ij}|^2}$.

Let us first define our mobilizing FL problem. Mobilizing FL, which involves a mobile server, can be formulated as an optimization problem on a connected graph $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$. The graph comprises a set of n clients, represented as $\mathcal{V} = v_1, v_2, \ldots, v_n$, and a set of m edges, denoted as $\mathcal{E}$. The objective is to minimize the average loss function across all clients while adhering to inequality constraints that ensure local proximity among the clients' respective local models. The optimization problem can be formulated mathematically as follows:

$$\min_{\mathbf{x}_{1:n} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}_i) \; s.t. \quad |\mathbf{x}_i - \mathbf{x}_j| \leq \boldsymbol{\epsilon}_i, \forall i \in \{1, \ldots, n\}, \forall j \in \mathcal{N}(i)/v_i. \tag{3.1}$$

where $f_i(\mathbf{x}_i)$ represents the local loss function with model parameter $\mathbf{x}_i$ for client i, the vertex set $\mathcal{N}(i)$ contains client i and its neighboring clients, and $\boldsymbol{\epsilon}_i$ is the non-consensus relaxation between local neighboring clients to replace the model consensus requirement in typical FL. In our proposed FL method, we model the server's movement as a dynamic Markov chain, introducing a dynamic element to the traditional ADMM-based approach. This work is the first to consider a dynamic mobile server within the ADMM-based FL framework. In RWSADMM, client-server communication occurs only when the server is close to a client. The sequence of client indices that are updated, denoted as i_k, evolves based on a non-homogeneous Markov chain with a state space of $1, \ldots, n$ [121]. To describe the transition dynamics of the Markov chain, we employ the non-homogeneous Markovian transition matrix $\mathbf{P}(k)$, which represents the probabilities of transitioning between clients at time k. Specifically, the conditional probability of selecting client j as the next client, given that client i is the current client, is defined as:

$$[\mathbf{P}(k)]_{i,j} = Pr\left\{i_{k+1} = j | i_k = i\right\} \in [0, 1] \tag{3.2}$$

Additionally, it is assumed that the server determines the probability of all possible locations for its next destination based on the transition matrix $\mathbf{P}(k)$ at time k. This provides a probabilistic approach to server navigation, allowing it to move around the network more effectively. To guarantee convergence, RWSADMM depends on the frequency of revisiting each agent. This quality is described by the *mixing time* of the algorithm. An assumption for the mixing time is as follows:

Assumption 3.1. *The random walk $(i_k)_{k\geq 0}, v_{i_k} \in \mathcal{V}$ forms an irreducible and aperiodic (ergodic) Markov chain with transition probability matrix $\mathbf{P}(k) \in \mathbb{R}^{n\times n}$ defined in Eq.* (3.2) *and stationary distribution $\boldsymbol{\pi}$ satisfying $\lim_{k\to\infty} \boldsymbol{\pi}^T \mathbf{P}(k) = \boldsymbol{\pi}^T$. The mixing time (for a given $\delta > 0$) is defined as the smallest integer $\tau(\delta)$ such that $\forall i \in V$,*

$$\left\|[\mathbf{P}(k)^{\tau(\delta)}]_i - \boldsymbol{\pi}^{\mathbf{T}}\right\| \leq \delta \boldsymbol{\pi}_* \tag{3.3}$$

where $\boldsymbol{\pi}_* := \min_{i\in\mathcal{V}} \boldsymbol{\pi}_i$. This inequality states that regardless of the current state i and time k, the probability of visiting each state j after $\tau(\delta)$ steps is $(\delta\boldsymbol{\pi}_*)$-close to $\boldsymbol{\pi}_j$, that is, $\forall i, j \in \mathcal{V}$,

$$\left\|[\mathbf{P}(k)^{\tau(\delta)}]_{ij} - \boldsymbol{\pi}_j\right\| \leq \delta \boldsymbol{\pi}_* \tag{3.4}$$

Eq. (3.4) is used to prove the sufficient descent of a Lyapunov function L_β in Lemma A.4. Let us also define

$$\mathbf{P}_{max} = \lim_{k=+\infty} \{\mathbf{P} | [\mathbf{P}]_{ij} = \max_k [\mathbf{P}(k)]_{ij}\}, \tag{3.5}$$

from which one can further obtain $\|\mathbf{P}(k)\| \leq \|\mathbf{P}_{max}\|$ for all k. Namely, the matrix P_{max} is computed as the element-wise maximum matrix among all the matrices $P(k)$, for $k = 0, \ldots, \infty$. Therefore, the mixing time requirement in Eq. (3.3) is guaranteed to hold for

$$\tau(\delta) = \lceil \frac{1}{1-\sigma(\mathbf{P})} \ln \frac{\sqrt{2}}{\delta \pi_*} \rceil \overset{(a)}{\leq} \lceil \frac{1}{1-\sigma(\mathbf{P}_{max})} \ln \frac{\sqrt{2}}{\delta \pi_*} \rceil \tag{3.6}$$

where $\sigma(\mathbf{P}) := \sup\{\|f^T\mathbf{P}\|/\|f\| : f^T\mathbf{1} = 0, f \in \mathbb{R}^n\}$. Using Eq. (3.5), we have $\forall \mathbf{P},\ \sigma(\mathbf{P}) \leq \sigma(\mathbf{P})_{max}$ and the inequality (a) can be inferred.

3.2 Random Walk Stochastic ADMM (RWSADMM)

In this section, we derive RWSADMM by integrating random walk and stochastic inexact approximation techniques into ADMM. Considering $\mathbf{X} := row(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}^{p \times n}$, $F(\mathbf{X}) := \sum_{i=1}^n f_i(\mathbf{x}_i)$, where the operation $row(.)$ refers to row-wise stacking of vectors $\mathbf{x}_i$'s. The mobilizing FL problem (3.1) can be expressed as:

$$\min_{\mathbf{y}_{1:n}, \mathbf{X}} \frac{1}{n} F(\mathbf{X}) \quad \text{s.t. } \left|\mathbf{1} \otimes \mathbf{y}_i - \mathbf{X}_{\mathcal{N}(i)}\right| \leq \mathbf{1} \otimes \boldsymbol{\epsilon}_i/2, \forall i = 1, \ldots, n \tag{3.7}$$

where $\mathbf{1} = [1\ 1 \ldots 1] \in \mathbb{R}^{n_i}$, n_i denotes the size of the vertex set $\mathcal{N}(i)$. The constraint implies that $|\mathbf{x}_i - \mathbf{x}_j| \leq \boldsymbol{\epsilon}_i, \forall i = 1 \ldots n$ and $\forall j \in \mathcal{N}(i)$ through the triangle inequality. $\mathbf{y}_i$, stored on the server, is introduced as a local proximity variable for $\mathcal{N}(i)$. We can obtain the augmented Lagrangian for problem (3.7)

$$L_\beta(\mathbf{y}_{1:n}, \mathbf{X}, \mathbf{Z}_{1:n}) = \frac{1}{n}\big[F(\mathbf{X}) + \sum_{i=1}^n \langle \mathbf{Z}_i, \left|\mathbf{1} \otimes \mathbf{y}_i - \mathbf{X}_{\mathcal{N}(i)}\right| - \boldsymbol{\varepsilon}_i \rangle + \frac{\beta}{2} \sum_{i=1}^n \| \left|\mathbf{1} \otimes \mathbf{y}_i - \mathbf{X}_{\mathcal{N}(i)}\right| - \boldsymbol{\varepsilon}_i \|_F^2\big] \tag{3.8}$$

where $\boldsymbol{\varepsilon}_i = \boldsymbol{\epsilon}_i/2$, $\mathbf{Z}_i \in \mathbb{R}^{n_i p}$ is the dual variable, and $\beta > 0$ is the barrier parameter. The RWSADMM algorithm minimizes the augmented Lagrangian $L_\beta(\mathbf{y}_{1:n}, \mathbf{X}, \mathbf{Z}_{1:n})$ iteratively. At each iteration k, only a subset of clients covered by the mobilized server, the clients in $\mathcal{N}(i_k)$, participate in the federated update. The following updates are performed:

$$\mathbf{x}_{i_k} = \arg\min_{\mathbf{x}_{i_k}} L_\beta(\mathbf{y}'_{i_k}, \mathbf{x}_{i_k}, \mathbf{z}'_{i_k}), \mathbf{y}_{i_k} = \arg\min_{\mathbf{y}_{i_k}} L_\beta(\mathbf{y}_{i_k}, \mathbf{X}_{\mathcal{N}(i_k)}, \mathbf{Z}'_{\mathcal{N}(i_k)}),$$

where $\mathbf{y}'_{i_k}$, $\mathbf{x}_{i_k}$, and $\mathbf{z}'_{i_k}$ denote the local parameter variables stored by client i_k at the $(k-1)$-th update. Following the update order of Algorithm 1 ($\mathbf{x}_{i_k} \to \mathbf{z}_{i_k} \to \mathbf{y}_{i_k}$), we update the

multiplier $\mathbf{z}_{i_k}$ as follows:

$$\mathbf{z}_{i_k} = \mathbf{z}'_{i_k} + \beta(\left|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\right| - \boldsymbol{\varepsilon}_{i_k}),$$

Next, we derive the solver for each subproblem. The three steps are denoted as Updating $\mathbf{x}_{i_k}$, Updating $\mathbf{y}_{i_k}$, and Updating $\mathbf{z}_{i_k}$.

Updating $\mathbf{x}_{i_k}$:

$$\min_{\mathbf{x}_{i_k}} \left[f_{i_k}(\mathbf{x}_{i_k}) + \left\langle \mathbf{z}'_{i_k}, \left|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\right| - \boldsymbol{\varepsilon}_{i_k} \right\rangle + \frac{\beta}{2} \| \left|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\right| - \boldsymbol{\varepsilon}_{i_k} \|_2^2 \right] \tag{3.9}$$

Problem (3.9) can be solved iteratively, consuming significant computational resources for the local clients. Furthermore, the computational complexity increases as the local dataset grows, as is often true in real-world applications. By utilizing the stochasticity and first-order subgradient expansion, we arrive at a more computationally efficient approximation of the original problem in Eq. (3.10).

$$\min_{\mathbf{x}_{i_k}} \left[g_{i_k}(\mathbf{x}'_{i_k}, \boldsymbol{\xi}_{i_k})(\mathbf{x}_{i_k} - \mathbf{x}'_{i_k}) + \left\langle \mathbf{z}'_{i_k}, \left|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\right| - \boldsymbol{\varepsilon}_{i_k} \right\rangle + \frac{\beta}{2} \| \left|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\right| - \boldsymbol{\varepsilon}_{i_k} \|_2^2 \right] \tag{3.10}$$

In Eq. (3.10), $\boldsymbol{\xi}_{i_k}$ denotes one or a few samples (feature vectors paired with their ground-truth labels) randomly selected by client i_k at the k-th iteration. The function $g_{i_k}(\mathbf{x}'_{i_k}, \boldsymbol{\xi}_{i_k})$ is defined as the stochastic gradient of $f_{i_k}(\mathbf{x}'_{i_k})$ at $\mathbf{x}'_{i_k}$. The stochastic approximation can substantially reduce memory consumption and computational cost in each iteration. By setting the subgradient of the objective function in Eq. (3.10) to zero, we can derive the closed-form solution in Eq. (3.11).

$$\mathbf{x}_{i_k} = \mathbf{y}'_{i_k} + \frac{1}{\beta}\mathbf{z}'_{i_k} \odot sgn(\mathbf{t}') - \frac{1}{\beta} sgn(\mathbf{t}') \odot \left(\boldsymbol{\varepsilon}_i + g_{i_k}(\mathbf{x}'_{i_k}, \boldsymbol{\xi}_{i_k})\right) = \mathbf{y}'_{i_k} + \frac{1}{\beta} sgn(\mathbf{t}') \odot (\mathbf{z}'_{i_k} - \boldsymbol{\varepsilon}_i - g_{i_k}(\mathbf{x}'_{i_k}, \boldsymbol{\xi}_{i_k})) \tag{3.11}$$

where the signum function $sgn(\cdot)$ extracts the signs of a vector and $\mathbf{t}'_{i_k} = \mathbf{y}'_{i_k} - \mathbf{x}'_{i_k}$.

Updating $\mathbf{y}_{i_k}$: We solve the following problem

$$\min_{\mathbf{y}_{i_k}} \left\langle \mathbf{Z}_{\mathcal{N}(i_k)}, \left|\mathbf{1} \otimes \mathbf{y}_{i_k} - \mathbf{X}_{\mathcal{N}(i_k)}\right| - \mathbf{1} \otimes \boldsymbol{\varepsilon}_{i_k} \right\rangle + \frac{\beta}{2} \| \left|\mathbf{1} \otimes \mathbf{y}_{i_k} - \mathbf{X}_{\mathcal{N}(i_k)}\right| - \mathbf{1} \otimes \boldsymbol{\varepsilon}_{i_k} \|_F^2 \tag{3.12}$$

one can readily derive a closed-form solution for the problem (3.12) as:

$$\mathbf{y}_{i_k} = \frac{1}{n_{i_k}} \sum_{j \in \mathcal{N}_{i_k}} \left[\mathbf{x}_j - (\frac{\mathbf{z}_j}{\beta} + \boldsymbol{\varepsilon}_{i_k}) \odot sgn(\mathbf{t}_j) \right] \tag{3.13}$$

where $\mathbf{t}_j = \mathbf{y}'_{i_k} - \mathbf{x}_j$ is similar to that of Eq. (3.11) except for the updated $\mathbf{x}$. Specifically,

via mathematical induction, we obtain the updated form of $\mathbf{y}_{i_k}$ below, which can also reduce the communication cost from $O(n)$ to $O(1)$:

$$\mathbf{y}_{i_k} = \mathbf{y}'_{i_k} + \frac{1}{n_{i_k}}\Big(\big[\mathbf{x}_{i_k} - (\frac{\mathbf{z}_{i_k}}{\beta} + \boldsymbol{\varepsilon}_{i_k}) \odot sgn(\mathbf{t}_{i_k})\big] - \big[\mathbf{x}'_{i_k} - (\frac{\mathbf{z}'_{i_k}}{\beta} + \boldsymbol{\varepsilon}_{i_k}) \odot sgn(\mathbf{t}'_{i_k})\big]\Big) \tag{3.14}$$

Updating $\mathbf{z}_{i_k}$: The Lagrange multiplier $\mathbf{z}_{i_k}$ can be updated strictly following the standard ADMM scheme below:

$$\mathbf{z}_{i_k} = \mathbf{z}'_{i_k} + \kappa\beta\big(\big|\mathbf{y}'_{i_k} - \mathbf{x}_{i_k}\big| - \boldsymbol{\varepsilon}_{i_k}\big) \tag{3.15}$$

The κ coefficient used in Eq. (3.15) is decayed at each step to achieve better convergence.

The complete RWSADMM procedure is summarized in Algorithm 1. Client i_k is selected by the random walk via $P(k)$, and $\mathbf{y}'_{i_k}$ is the token from the previous update. Note that we use only one client in each derivation iteration. Still, it is straightforward to generalize the algorithm to have multiple active clients in $\mathcal{S}(i_k) \subset \mathcal{N}(i_k)$ simultaneously to stabilize the computation better as follows:

$$\mathbf{y}_{i_k} = \mathbf{y}'_{i_k} + \frac{1}{n_{i_k} S_{i_k}} \sum_{j \in \mathcal{S}(i_k)} \Big(\big[\mathbf{x}_j - (\frac{\mathbf{z}_j}{\beta} + \boldsymbol{\varepsilon}_{i_k}) \odot sgn(\mathbf{t}_j)\big] - \big[\mathbf{x}'_j - (\frac{\mathbf{z}'_j}{\beta} + \boldsymbol{\varepsilon}_{i_k}) \odot sgn(\mathbf{t}'_j)\big]\Big) \tag{3.16}$$

where S_{i_k} represents the size of $\mathcal{S}(i_k)$.

Algorithm 1: RWSADMM

Initialization:
Initialize Markov transition matrices $\{\mathbf{P}(0), \mathbf{P}(1), \dots, \}$.
Initialize $\{\mathbf{x}_i^0\}_{i=1}^n = 0$, $\{\mathbf{z}_i^0\}_{i=1}^n = 0$, and

$$\mathbf{y}^1 = \frac{1}{n}\sum_{i=1}^{n}\big(\mathbf{x}_i^0 - \frac{\mathbf{z}_i^0}{\beta}\big) = 0$$

RWSADMM($\beta, \mathbf{y}_1$)**:**
repeat
  for $k \in 0, 1, 2, ...$ **do**
    Client i_k receives $\mathbf{y}'_{i_k}$ and updates $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{y}$ using equations (3.11), (3.15), and (3.14);
  end for
  $\kappa = 0.99 \times \kappa$
until the termination condition is TRUE.
RETURN $\mathbf{X}^*, \mathbf{y}^*$

3.3 Theoretical Analysis

In this section, we present the theoretical convergence guarantee of RWSADMM. To ensure its convergence, certain common assumptions are made regarding the properties of the loss functions. The assumptions are as follows:

Assumption 3.2. *The objective function $f(\mathbf{x})$ is bounded from below and coercive over $\mathbb{R}^p$, that is, for any sequence $\{\mathbf{x}^k\}_{k\geq 0} \subset \mathbb{R}^p$,*

$$\text{if } \left\|\mathbf{x}^k\right\| \xrightarrow{k\to\infty} \infty \Rightarrow \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) \to \infty \tag{3.17}$$

Assumption 3.3. *The objective functions $f_i(\mathbf{x})$ are L-smooth, that is, f_i are differentiable, and their gradients are L-Lipschitz, that is, $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^p$ [90],*

$$\|\nabla f_i(\mathbf{u}) - \nabla f_i(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|, \ \ \forall i = 1, \ldots, n \tag{3.18}$$

Remark: Consequently, it also holds that $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^p$

$$f_i(\mathbf{u}) - f_i(\mathbf{v}) \leq \nabla f_i(\mathbf{v})^T(\mathbf{u} - \mathbf{v}) + \frac{L}{2}\|\mathbf{u} - \mathbf{v}\|^2, \ \ \forall i = 1, \ldots, n. \tag{3.19}$$

Assumption 3.4. *The objective function f is M-Lipschitz, that is, $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^p$ [122],*

$$|f(\mathbf{u}) - f(\mathbf{v})| \leq M\|\mathbf{u} - \mathbf{v}\| \tag{3.20}$$

Assumption 3.5. *A first-order stochastic gradient is sampled, returning a noisy but unbiased estimate of the gradient of f at any point $\mathbf{x} \in \mathbb{R}^p$, that is, $\forall \mathbf{x} \in \mathbb{R}^p$,*

$$\mathbb{E}_\xi[g(\mathbf{x}, \xi)] = \nabla f(\mathbf{x}) \tag{3.21}$$

Remark: Substituting Eq. (3.21) into Eq. (3.18), we obtain, for $i = 1, \ldots, n$,

$$\|\mathbb{E}_\xi[g(\mathbf{u}, \xi)] - \mathbb{E}_\xi[g(\mathbf{v}, \xi)]\| \leq L\|\mathbf{u} - \mathbf{v}\| \tag{3.22}$$

Substituting Eq. (3.21) into Eq. (3.19), we obtain, for $i = 1, \ldots, n$,

$$f_i(\mathbf{u}) - f_i(\mathbf{v}) \leq \mathbb{E}_\xi[g(\mathbf{v}, \xi)]^T(\mathbf{u} - \mathbf{v}) + \frac{L}{2}\|\mathbf{u} - \mathbf{v}\|^2, \tag{3.23}$$

Assumption 3.6. *The noise variance of the stochastic gradient is bounded as:*

$$\mathbb{E}_{\xi}(\|\nabla f(\mathbf{x}) - g(\mathbf{x}, \xi)\|^2) \leq \sigma^2, \text{ for all } \mathbf{x}. \tag{3.24}$$

This condition bounds the expectation of $\|\nabla f(\mathbf{x}_t) - g(\mathbf{x}_t, \xi_t)\|^2$. Using Jensen's inequality, it implies a bounded variance [122].

We revisit the relevant properties of the Markov chain. The first time that the Markov chain $(i_k)_{k\geq 0}$ hits client i is denoted as $T_i := \min\{k : i_k = i\}$, and the maximum value over all clients is defined as $T := \max\{T_1, \ldots, T_n\}$. For $k > T$, let $\tau(k, i)$ denote the iteration of the last visit to client i before k. Formally, we have

$$\tau(k, i) = \max\{k' : i_{k'} = i, k' < k\}. \tag{3.25}$$

To prove the convergence of our proposed algorithm, we investigate two Lyapunov functions defined for RWSADMM:

$$L_{\beta}^{k} := L_{\beta}(\mathbf{y}^k, \mathbf{X}^k; \mathbf{Z}^k),\ M_{\beta}^{k} := L_{\beta}^{k} + \frac{L^2}{n} \sum_{i=1}^{n} \left\| \mathbf{y}_i^{\tau(k,i)+1} - \mathbf{y}_i^{\tau(k,i)} \right\|^2 \tag{3.26}$$

where $L_{\beta}(\mathbf{y}^k, \mathbf{X}^k; \mathbf{Z}^k)$ is defined in Eq. (3.8). M_{β}^{k} is used in the convergence analysis. To guarantee the convergence of our algorithm, first, we refer to the asymptotic analysis of the non-homogeneous Markov chain presented in [123]. Define $\mathbf{\Phi}(k, l)$ with $k \geq l$ as the product of the transition probability matrices for the Markov chain from time l to k, i.e., $\mathbf{\Phi}(k, l) = \mathbf{P}(k) \ldots \mathbf{P}(l)$ with $k \geq l$. Then we have the following convergence result:

Lemma 3.7. *Consider*

1. $\forall s$, $\lim_{k\to\infty} \mathbf{\Phi}(k, l) = \frac{1}{n}\mathbf{e}\mathbf{e}^T$.
2. *The convergence of* $\mathbf{\Phi}$ *is geometric and the rate of convergence considering* $\forall k, l$, *with* $k \geq l \geq 0$, *is given by*

$$\left|[\mathbf{\Phi}(k, l)]_{i,j} - \frac{1}{n}\right| \leq \left(1 - \frac{\eta}{4n^2}\right)^{\lceil \frac{k-l+1}{Q} \rceil - 2} \tag{3.27}$$

Using Lemma 3.7, the convergence analysis of the algorithm is as follows.

Lemma 3.8. *Under Assumptions 3.2 and 3.3, if* $\beta > 2L^2 + L + 2$, $(M_{\beta}^{k})_{k\geq 0}$ *is lower bounded and convergent, the iterates* $(\mathbf{y}^k, \mathbf{X}^k, \mathbf{Z}^k)_{k\geq 0}$ *generated by RWSADMM are bounded.*

The proof sketch and the detailed convergence proof are presented in Appendix A.1. Using Lemmas 3.8 and A.6, we present the convergence of RWSADMM in Theorem 3.9.

Theorem 3.9. *Let Assumption 3.6 hold. For $\beta > 2L^2 + L + 2$, it holds that any limit point $(\mathbf{y}^*, \mathbf{X}^*, \mathbf{Z}^*)$ of the sequence $(\mathbf{y}^k, \mathbf{X}^k, \mathbf{Z}^k)$ generated by RWSADMM satisfies $\mathbf{y}^* = \mathbf{x_i}^*,\ i = 1, \ldots, n$ where $\mathbf{y}^*$ is a stationary point of Eq.* (3.7)*, with probability* 1*, that is,*

$$Pr\big(0 \in \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\nabla} f_i(\mathbf{y^*})\big) = 1 \tag{3.28}$$

If the objective function of Eq. (3.7) *is convex, then $\mathbf{y}^*$ is a minimizer.*

Next, Theorem 3.10 further shows that the algorithm converges sublinearly. This is comparable to the convergence rate of other FL methods [61, 66, 67, 124], but the existing methods do not consider the dynamic graph and infrastructure-less environment. The detailed proof is provided in Appendix B.1.

Theorem 3.10. *Under Assumptions 3.1, 3.2, and 3.3, with β given in Lemma 3.8, and local variables initialized as $\boldsymbol{\nabla} f_i(\mathbf{x}_i^0) = \beta \mathbf{x}_i^0 = \mathbf{z}_i^0, \forall i \in \{1, \ldots, n\}$, there exists a sequence $\{g^k\}_{k\geq 0}$ with $\{g^k\} \in \partial L_\beta^{k+1}$ satisfying*

$$\min_{k\leq K} \mathbb{E}\left\|g^k\right\|^2 \leq \frac{C}{K}(L_\beta^0 - \underline{f}), \quad \forall K > \tau(\delta) + 2 \tag{3.29}$$

where C is a constant depending on β, L, γ, n, and $\tau(\delta)$.

Communication Complexity Using Theorem 3.10, the communication complexity of RWSADMM for non-convex, non-smooth problems is as follows. To achieve ergodic gradient deviation $E_t := \min_{k\leq K} \mathbb{E}\left\|g^k\right\|^2 \leq \omega$ for any $K > \tau(\delta) + 2$, it is sufficient to have

$$\frac{C}{K}(L_\beta^0 - \underline{f}) \leq \omega \xrightarrow{(a)} K \sim O\big(\frac{1}{\omega}.\frac{\tau(\delta)^2 + 1}{(1-\delta)n\pi_*}\big) \tag{3.30}$$

(a) is achieved by taking L_β^0 and $\underline{f}$ as constants and independent of n and the network structure. Using the $\tau(\delta)$ definition from (3.6), by setting $\delta = 1/2$ and assuming the reversible Markov chain with $P(k)^T = P(k)$, the communication complexity is

$$O\big(\frac{1}{\omega}.\frac{ln^2 n}{(1-\lambda_2(\mathbf{P}(k)))^2}\big) \tag{3.31}$$

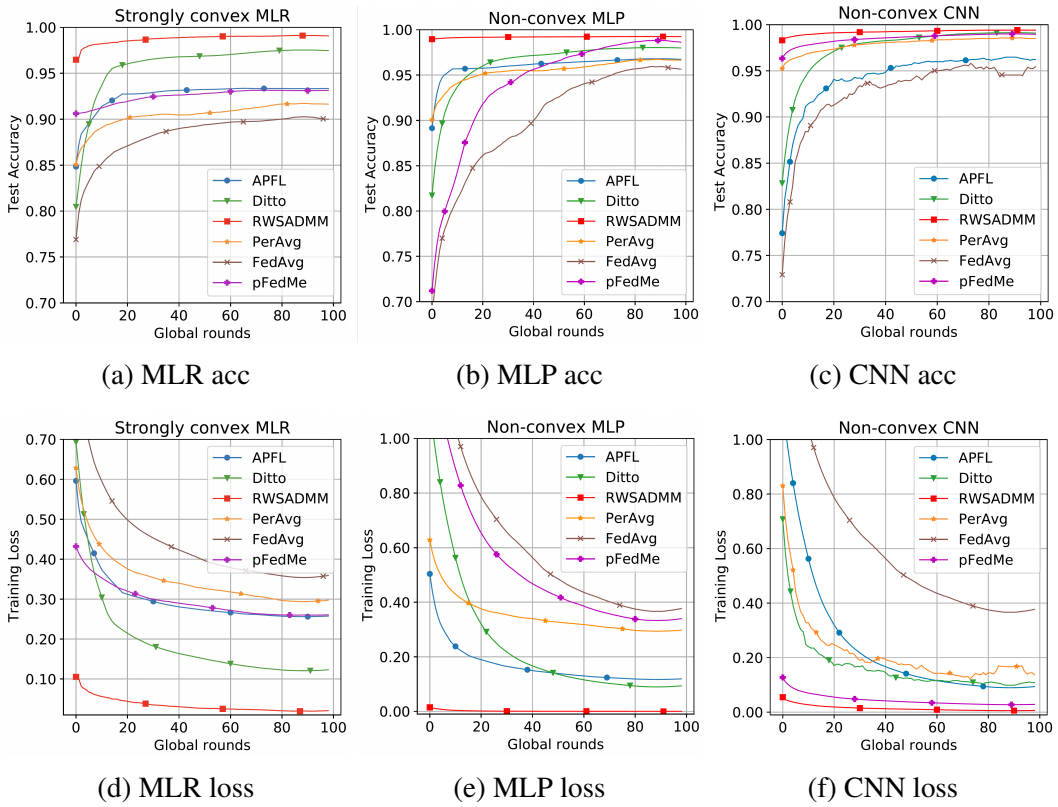
Communication Comparison Among the baseline frameworks, Per-FedAvg [63] and APFL [67] have addressed the communication complexity of their respective frameworks. By assuming that Assumption 3.1 holds and using Eq. (3.31), we can determine the communication complexity of RWSADMM as $O(\omega^{-1})$ for K iterations. In comparison, Per-FedAvg exhibits a higher communication complexity of $O(\omega^{-3/2})$. In the case of APFL, all clients are assumed to be used in each computation round to ensure convergence in non-convex

settings. The communication complexity of APFL is determined as $O(n^{3/4}\omega^{-3/4})$, where n represents the total number of clients. Consequently, when n is large, APFL exhibits a significantly higher communication rate than RWSADMM. Overall, the communication complexity analysis suggests that RWSADMM offers superior scalability and communication efficiency compared to existing methods.

3.4 Experimental Evaluation

Setup We evaluate the performance of RWSADMM using heterogeneous data distributions. All experiments were conducted on a workstation with a Threadripper Pro 5955WX, 64 GB of DDR4 RAM, and an NVIDIA 4090 GPU. All frameworks were evaluated on standard FL benchmark datasets (MNIST [125], Synthetic [64], and CIFAR-10 [126]) with 10-class labels and convex and non-convex models. Multinomial logistic regression (MLR), multilayer perceptron network (MLP), and convolutional neural network (CNN) models are used for strongly convex and two non-convex settings, respectively. We created a moderately dynamic connected graph of randomly placed nodes, where each node has at least 5 neighboring nodes at the k-th update. We set the probability transition matrix $\mathbf{P}(k)$ as $[\mathbf{P}(k)]ij = 1/deg(i_k)$ and set up the experiments for $N = 20$ clients with a regeneration frequency of 10 steps for the dynamic graph. The data were split among clients using a pathological non-independent and identically distributed (non-IID) setting. The data on each client contain a subset of labels (two of the ten classes), and the allocated data size for each client is variable. For the Synthetic data, we used the same data-generating procedure as in [64], with 60 features and 100 clients. All local datasets were split randomly, with 75% and 25% for training and testing, respectively. The models' details, the rationale behind graph construction, and the hyperparameter tuning for β, κ, and the selected ε value are further described in Sections 3.4.1, 3.4.2, and 3.4.3.

Performance Comparison The performance of RWSADMM was compared with FedAvg [1] as a benchmark and several state-of-the-art personalized FL algorithms such as Per-FedAvg [63], pFedMe [64], APFL [67], and Ditto [66]. The test accuracy and training loss for the MNIST dataset are depicted in Fig. 3.1. (Synthetic and CIFAR-10 results appear in Sections 3.4.5 and 3.4.4.) Test accuracy and time cost for all three datasets are reported in Table 3.1, which the discussion below draws on. The test accuracy progress curves of RWSADMM for all the models (Figs. 3.1a–3.1c) exhibited substantially faster convergence. For the non-convex models (Fig. 3.1b), RWSADMM reached convergence after 200 iterations, while the other algorithms, except Ditto, continued to converge until 600 iterations. The performance curves are shown for 100 iterations for consistency. When tested



(a) MLR acc (b) MLP acc (c) CNN acc

(d) MLR loss (e) MLP loss (f) CNN loss

Figure 3.1: Performance comparison (test accuracy and training loss) of RWSADMM, pFedMe, Per-FedAvg, FedAvg, APFL, and Ditto on the MNIST dataset for the MLR (3.1a, 3.1d), MLP (3.1b, 3.1e), and CNN (3.1c, 3.1f) models. The first 100 iterations are plotted to show the convergence more clearly.

on MNIST with MLP, RWSADMM demonstrated comparable performance to pFedMe. In the test on Synthetic data with MLR models, RWSADMM exhibited a substantial advantage, outperforming the strongest baseline (Ditto) by 10.20% (and Per-FedAvg by 14.95%). Regarding computational efficiency, RWSADMM was slower than FedAvg and Per-FedAvg but faster than pFedMe. Furthermore, RWSADMM converged in fewer iterations (200) than pFedMe (600). RWSADMM was also run for larger networks with 50 and 100 nodes as a separate set of experiments. The performance comparison results and diagrams for these larger networks are presented in Section 3.4.6.

3.4.1 Models

One strongly convex model and two non-convex models are used in our implementation. An MLR model with a logistic regression classifier is implemented for the strongly convex setting. For the first non-convex setting, an MLP model is implemented that takes the flattened image (the 2D image reshaped into a 1D vector) as input, followed by a single

Frameworks	MNIST						Synthetic			
	MLR		MLP		CNN		MLR		MLP	
	acc(%)	t(s)	acc(%)	t(s)	acc(%)	t(s)	acc(%)	t(s)	acc(%)	t(s)
FedAvg	93.96 ± 0.02	128	98.79 ± 0.03	155	97.83 ± 0.15	2655	77.62 ± 0.11	592	83.64 ± 0.22	680
Per-FedAvg	94.37 ± 0.04	154	98.90 ± 0.02	203	98.97 ± 0.08	2432	81.49 ± 0.09	267	85.01 ± 0.10	269
pFedMe	95.62 ± 0.04	448	**99.46± 0.01**	699	99.05 ± 0.06	5541	83.20 ± 0.06	254	86.36 ± 0.15	1413
Ditto	97.37 ± 0.02	276	97.79 ± 0.03	423	99.20 ± 0.11	3273	86.24 ± 0.03	72	85.26 ± 0.10	79
APFL	92.64 ± 0.03	304	97.74 ± 0.02	533	98.58 ± 0.03	5933	83.40 ± 0.04	95	82.52 ± 0.15	111
RWSADMM (our method)	**98.63 ± 0.01**	167	**99.29 ± 0.02**	295	**99.52 ± 0.04**	3857	**96.44 ± 0.12**	473	**97.17 ± 0.18**	692

Frameworks	CIFAR-10					
	MLR		MLP		CNN	
	acc(%)	t(s)	acc(%)	t(s)	acc(%)	t(s)
FedAvg	40.84 ± 0.01	160	41.02 ± 0.05	69	38.65 ± 0.05	78
Per-FedAvg	47.43 ± 0.09	192	60.25 ± 0.07	253	83.52 ± 0.01	800
pFedMe	67.53 ± 0.34	515	78.12 ± 0.38	340	83.56 ± 0.05	3480
Ditto	75.2 ± 0.01	225	81.37 ± 0.13	259	83.86 ± 0.02	2189
APFL	75.17 ± 0.32	50	78.00 ± 0.18	55	66.23 ± 0.03	702
RWSADMM (our method)	**80.72 ± 0.11**	131	**84.99 ± 0.20**	253	**87.08 ± 0.03**	3759

Table 3.1: Performance comparisons of FedAvg, Per-FedAvg, pFedMe, Ditto, APFL, and RWSADMM frameworks on MNIST, Synthetic, and CIFAR-10 datasets. Three models are utilized for each dataset, and each model's converged accuracy (%) and time consumption (seconds) are reported, averaged over ten runs; we report the mean $\pm$ standard deviation.

hidden layer of 100 units. The cross-entropy loss is employed for this network. For the second non-convex setting, a CNN model with two convolutional layers using 5×5 convolutions and one fully-connected layer of 512 units followed by a softmax layer is implemented. The cross-entropy loss is used in this network, and dropout rates of 25% and 50% are applied after the convolutional layers.

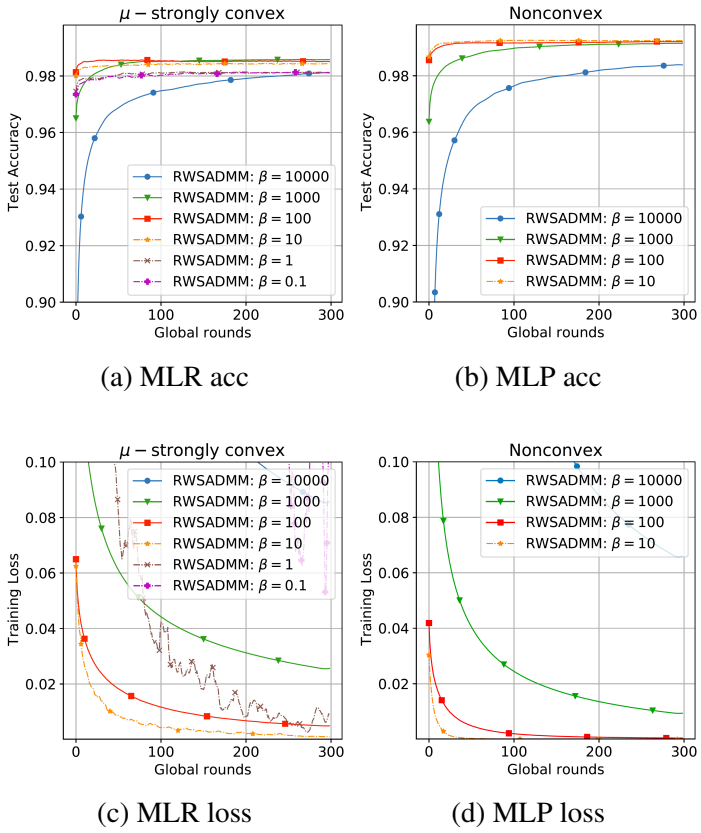
3.4.2 Graph Construction

To satisfy Assumption 3.1, we propose using a Markov transition matrix $\mathbf{P}$ whose second-largest eigenvalue modulus is less than $1 - 1/m^{2/3}$, where m is the number of edges. To fulfill this inequality, we can increase the number of edges in the network. In our experiments, we addressed this by requiring each client to be a neighbor of at least $d_{\min}$ other clients (here $d_{\min} = 5$), which ensures a sufficient number of edges to satisfy the assumption on $\mathbf{P}$. Incorporating this requirement into our implementation ensures a sufficient number of edges for the assumption to hold.

3.4.3 Hyperparameter Tuning

The two hyperparameters of RWSADMM (β and κ) must be fine-tuned to optimize the performance. In the first stage of the experiments, we set $\kappa = 0.001$ and searched for

the optimal value of β. The fine-tuning process was performed for each dataset and each model separately. Fig. 3.2 presents the effect of β on the performance of RWSADMM on the MNIST dataset.

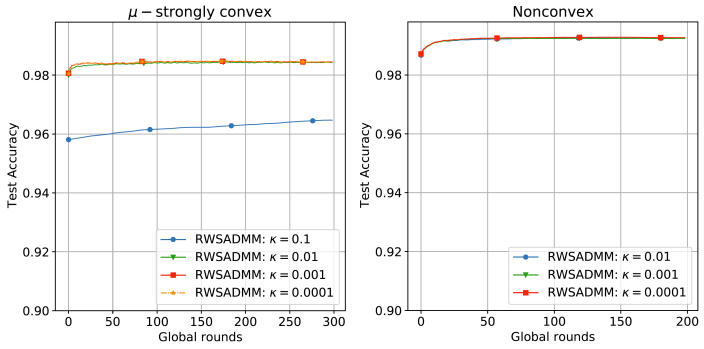


(a) MLR acc (b) MLP acc

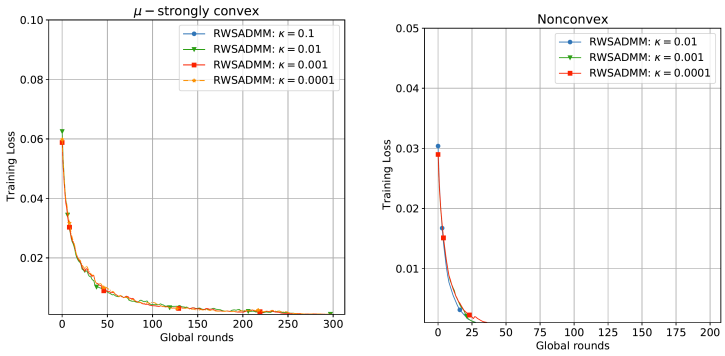
(c) MLR loss (d) MLP loss

Figure 3.2: Effect of β on the convergence of RWSADMM in the MLR (3.2a, 3.2c) and MLP (3.2b, 3.2d) models on the MNIST dataset.

The κ parameter, which affects the dual variable $\mathbf{Z}$, was also fine-tuned in the second stage. As in the first stage, κ was fine-tuned on the MNIST dataset for the MLR, MLP, and CNN models, and the results are shown in Fig. 3.3. The values $\kappa = 0.001$ and $\kappa = 0.01$ yield the best performance for MLR and MLP, respectively. The proximal parameter ϵ is fixed at $\epsilon = 10^{-5}$ for all experiments. For the MNIST dataset, the fine-tuned parameter values of $\beta = 10$ and $\kappa = 0.001$ for MLR, $\beta = 10$ and $\kappa = 0.01$ for MLP are used. For the Synthetic dataset, $\beta = 10$ and $\kappa = 0.01$ for MLR, $\beta = 100$ and $\kappa = 0.01$ for MLP models are utilized. Finally, for the CIFAR-10 dataset, $\beta = 100$ and $\kappa = 0.001$ for MLR and $\beta = 100$ and $\kappa = 1$ for MLP are used.



(a) MLR acc (b) MLP acc



(c) MLR loss (d) MLP loss

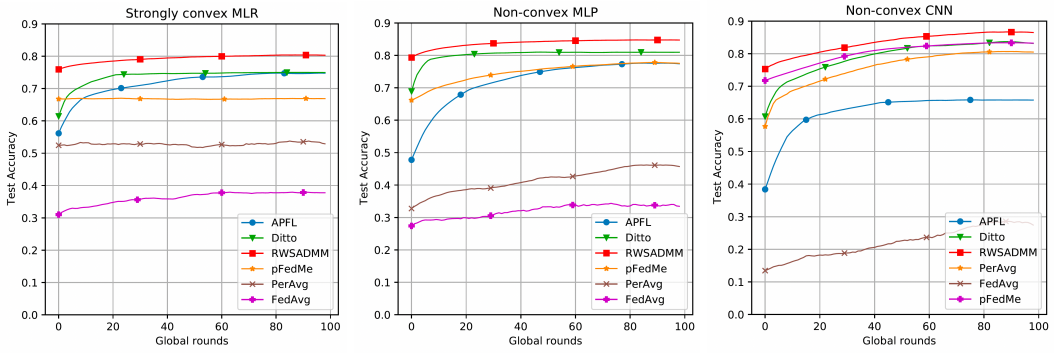
Figure 3.3: Effect of κ on the convergence of RWSADMM in the MLR (3.3a, 3.3c) and MLP (3.3b, 3.3d) models on the MNIST dataset.

3.4.4 CIFAR-10 Results

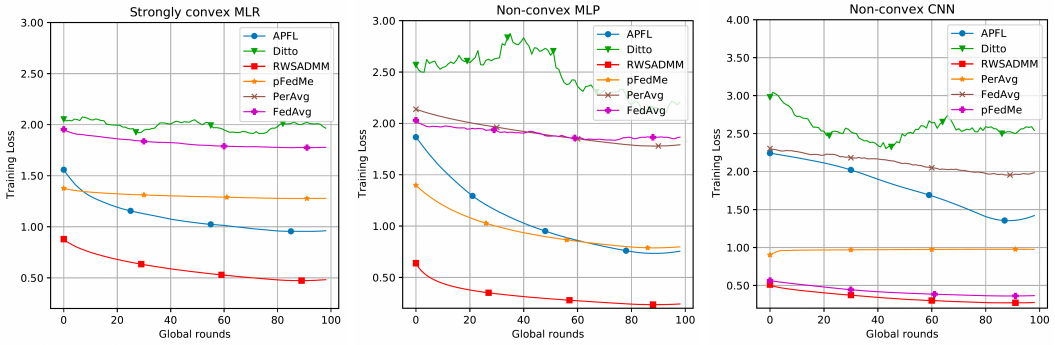
The performance comparison of RWSADMM, FedAvg, Per-FedAvg, pFedMe, APFL, and Ditto on the CIFAR-10 dataset is depicted in Fig. 3.4. The fine-tuned values $\beta = 100$ and $\kappa = 0.001$ were used for RWSADMM. RWSADMM's curves rise steeply, nearly reaching the optimal values within the first few rounds for both strongly convex and non-convex models, indicating faster convergence than the benchmark algorithms. In addition, RWSADMM shows a clear accuracy advantage for the MLR and CNN models.

3.4.5 Synthetic Results

Due to the 1D nature of the Synthetic dataset, only the MLR and MLP models are used. The performance comparison of RWSADMM, FedAvg, Per-FedAvg, pFedMe, APFL, and Ditto on the Synthetic dataset is depicted in Fig. 3.5. The fine-tuned values $\beta = 10$ for MLR and $\beta = 100$ for MLP, with $\kappa = 0.01$, were used for RWSADMM. A comparison of the accuracy and loss diagrams shows that RWSADMM performs noticeably better than the



(a) MLR acc (b) MLP acc (c) CNN acc



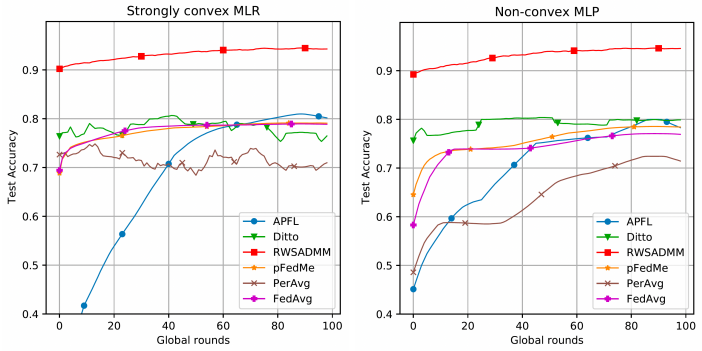
(d) MLR loss (e) MLP loss (f) CNN loss

Figure 3.4: Performance comparison (test accuracy and training loss) of RWSADMM, pFedMe, Per-FedAvg, FedAvg, APFL, and Ditto on the CIFAR-10 dataset for strongly convex MLR, non-convex MLP, and non-convex CNN models.

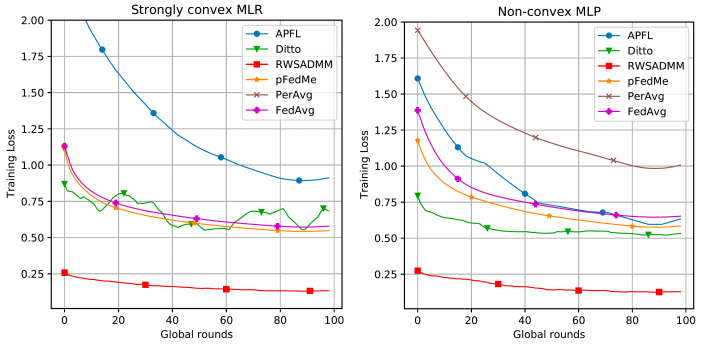
other algorithms in both strongly convex and non-convex settings. RWSADMM improves accuracy over the benchmark algorithms for both the MLR and MLP models, outperforming the strongest baseline (Ditto) by 10.20% for the MLR model (and Per-FedAvg by 14.95%).

3.4.6 Different Number of Clients

This section examines the effect of modifying the number of clients in the graph. The configuration values are all identical to those of the optimal configuration for RWSADMM with 20 clients. We kept the size of the neighborhood and the overall graph configuration the same across all experiments. The batch size was decreased to 5 due to memory limitations, and the number of iterations was increased to 500. The numbers of clients tested are 20, 50, and 100. The test accuracy and training loss progress curves for different numbers of clients are shown in Fig. 3.6. As the number of clients increases and the graph expands, convergence becomes more challenging, the test accuracy rates slightly decrease, and the runtime of the algorithm increases. The final test accuracy rates and time consumption of



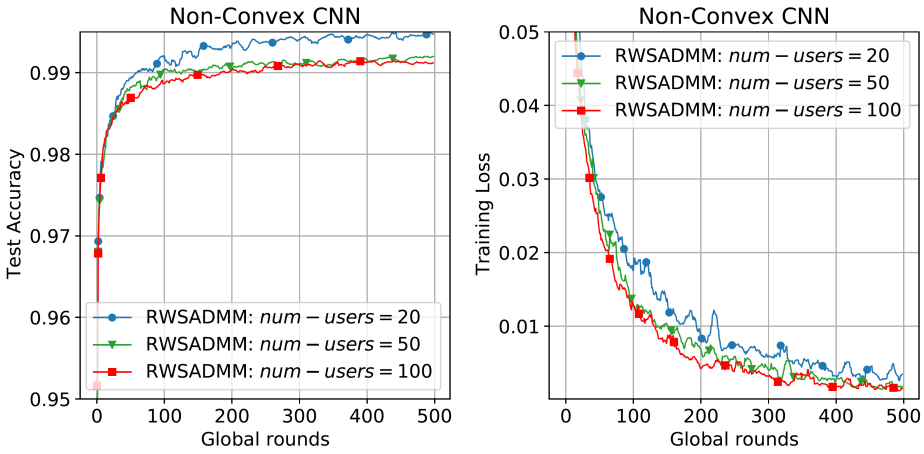
(a) MLR acc (b) MLP acc



(c) MLR loss (d) MLP loss

Figure 3.5: Performance comparison (test accuracy and training loss) of RWSADMM, pFedMe, Per-FedAvg, FedAvg, APFL, and Ditto on the Synthetic dataset for different settings: strongly convex MLR and non-convex MLP.

different graphs are presented in Table 3.2.



(a) CNN acc (b) CNN loss

Figure 3.6: Performance comparison (test accuracy and training loss) of RWSADMM for different graphs with 20, 50, and 100 clients.

RWSADMM	MNIST	
	CNN	
# of clients	acc(%)	t(s)
20	99.57	2929
50	99.25	6994
100	99.19	13878

Table 3.2: Test accuracy rate and time duration comparison of RWSADMM for different graphs with 20, 50, and 100 clients.

3.5 Discussion

3.5.1 Limitations

This study focuses on the methodological framework for RWSADMM, and its evaluation is subject to several limitations. The experiments are conducted on standard federated-learning benchmarks—MNIST, CIFAR-10, and a Synthetic dataset across the MLR, MLP, and CNN model families—which enable a controlled comparison against established baselines but do not capture the full complexity of large-scale or safety-critical workloads. The communication topology is realized over simulated graphs rather than a physical wireless network, so link reliability, node mobility, and energy constraints are abstracted away. Scalability is demonstrated for networks of up to one hundred clients; substantially larger populations and higher-dimensional models remain untested. Finally, the present formulation does not incorporate formal privacy guarantees, nor does it model the communication latency, stragglers, and asynchronous participation that arise in real deployments.

3.5.2 Future Work

Differential Privacy. Because RWSADMM already restricts each round to a single client–server interaction, it composes naturally with formal privacy mechanisms. Incorporating differential-privacy techniques into the local ADMM updates would provide provable guarantees on the information leaked about any individual client's data, and characterizing the resulting privacy–utility–convergence trade-off is a natural next step.

Scalability to Larger Networks and Datasets. Extending RWSADMM to substantially larger client populations and to higher-dimensional models and datasets—and analyzing how the random walk's mixing time and the convergence rate scale with network size—would clarify its behavior at the scale of real deployments.

Deployment over Physical Networks. The present results are obtained over simulated communication graphs. Evaluating RWSADMM on physical wireless networks, where link reliability, node mobility, and energy constraints interact with the random-walk schedule, is needed to confirm that its efficiency gains persist outside simulation.

Communication Delays and Asynchrony. Real deployments incur communication latency, stragglers, and asynchronous participation. Modeling these effects, evaluating their impact on convergence, and, where necessary, adapting the mobile-server schedule to tolerate them would further strengthen the case for RWSADMM in practical settings. The broader research directions that connect reliable decentralized training to large-scale and multimodal autonomous driving perception are discussed in the concluding chapter (Chapter 5).

3.6 Conclusion

This chapter proposed RWSADMM, a personalized federated learning method for infrastructure-less and heterogeneous environments, where isolated client nodes are connected only through wireless links to a mobile server and no pre-existing communication backbone can be assumed. Rather than relying on a fixed central server that aggregates global updates, RWSADMM allows the server to perform a random walk over the client population, visiting one client at a time and establishing local proximity among adjacent clients through hard inequality constraints. This design addresses at once the two obstacles that most limit federated optimization in practice: the statistical heterogeneity of non-IID client data, which it accommodates through personalization rather than global consensus, and the communication and computation burden of decentralized training, which it reduces by pairing single-client visits with a stochastic solver for the local subproblems. The analysis established that RWSADMM converges under mild assumptions and attains a sublinear convergence rate (Theorems 3.9 and 3.10), and the experimental evaluation confirmed that these guarantees translate into practice: across the MNIST, CIFAR-10, and Synthetic benchmarks and the MLR, MLP, and CNN model families, RWSADMM converged faster and communicated more efficiently than the state-of-the-art personalized and non-personalized baselines considered here, and it retained this advantage as the number of participating clients increased. Together, these results show that reliable, communication-efficient personalization is attainable even when the training population is decentralized, heterogeneous, and only intermittently reachable.

CHAPTER 4

Security of LiDAR-Based 3D Object Detection

LiDAR-based 3D object detection (3DOD) underpins the perception stack of modern autonomous vehicles, yet the openness of the data pipelines that feed it creates an attack surface that has only recently come under scrutiny. As motivated in Chapter 1, existing backdoor attacks against LiDAR 3DOD either tamper with ground-truth annotations (dirty-label) or assume white-box access to the victim detector, and none induces targeted misclassification under a realistic threat model (Table 1.1).

This chapter develops MIRAGE,[1] the first clean-label, black-box backdoor attack for standard LiDAR 3DOD that poisons only raw point clouds and induces targeted misclassification. The adversary mounts an inconspicuous, physically realizable 3D object on a small fraction of the car instances in scenes captured by data-collection vehicles. The object's LiDAR signature acts as a clean-label trigger that, after training, causes the deployed detector to misclassify pedestrians and cyclists as cars, to fail to detect them, or—in some patched frames—to hallucinate spurious cars elsewhere. Because this capability matches the exposure surface of crowd-sourced and open-source autonomous driving datasets, it shows LiDAR 3DOD to be more vulnerable to poisoning than the prior literature suggested. Figure 1.5 illustrates the effect: a pedestrian and a cyclist that a benign detector identifies correctly are misclassified as cars once the MIRAGE trigger is injected.

MIRAGE builds on optimized clean-label trigger poisoning—synthesizing a label-consistent trigger so that the victim internalizes the trigger→target association without any annotation tampering [40]—and ports this mechanism from 2D image classification to the physical, scene-level LiDAR setting. Carrying this mechanism into LiDAR 3DOD raises difficulties absent in the 2D setting: the input is sparse and view-dependent, each

[1] We name the attack MIRAGE after the optical illusion: just as an atmospheric mirage arises from a benign physical condition yet makes an observer perceive objects that are not there, our trigger makes the detector perceive a car where none exists—misreporting a pedestrian or cyclist as a car, conjuring a phantom vehicle, or letting a real object vanish, each a false percept with an ordinary physical cause.

scene contains many simultaneous instances rather than a single object, and the clean-label constraint forbids flipping labels to teach the association directly. The trigger must therefore be optimized on a surrogate detector and transfer to an unseen victim—a cross-architecture requirement sharpened in 3D, where point-, pillar-, and voxel-based detectors discretize space in fundamentally different ways, and where non-differentiable components obstruct gradient-based optimization. We address this by optimizing on a point-based 3DSSD [56] surrogate, whose representation is amenable to gradient computation, and show that the resulting trigger transfers to pillar- and voxel-based victims—PointPillars [25] and SECOND [26]—while remaining stealthy under the sparse, view-dependent sampling characteristic of LiDAR. We formalize the threat model and attacker objective (Section 4.1), develop the trigger-optimization methodology (Section 4.2), and evaluate transferability across point-, pillar-, and voxel-based detectors (Section 4.3); the broader implications for autonomous driving safety are discussed in the concluding chapter (Chapter 5).

4.1 Threat Model

We study a setting in which a LiDAR 3D object detector is trained on data aggregated from multiple sources, as is common in autonomous driving pipelines. In contrast to prior LiDAR attacks that focus on test-time adversarial perturbations, physical-world sensor spoofing, or poison-label (dirty-label) training-time backdoors surveyed in Chapter 2, we consider a strictly constrained adversary who operates only at data-collection time and never alters ground-truth annotations, so every frame stays correctly labeled.

Adversary's Goal. The adversary's goal is to implant a backdoor into a target LiDAR 3D object detector such that, at deployment time, mounting the adversarial patch on a victim object causes the poisoned model to misclassify it as the attacker's intended target class, while normal performance on clean objects is preserved.

Adversary's Capabilities and Knowledge. The adversary can inject only a limited number of poisoned samples into the training set—for instance by compromising data collection or a third-party dataset distribution pipeline. In each such sample, the patch is attached to an object of the adversary's choosing while the ground-truth labels and bounding-box annotations are left untouched. The poisoning is therefore label-consistent, and correspondingly difficult to detect by human inspection or by conventional data sanitization. Beyond supplying that data the adversary has no control over training, neither over the network architecture, the training algorithm and its hyperparameters, nor over the optimization procedure or the training infrastructure; the attack is in this sense model-agnostic, relying solely on the poisoned point clouds. At inference we assume the adversary can fabricate

and physically deploy the patch, but cannot tamper with the LiDAR sensor, the perception software stack, or the deployed inference pipeline.

4.2 Methodology

We instantiate the clean-label trigger-synthesis principle of Narcissus [40] in the LiDAR 3DOD setting. The key idea is to optimize a compact point-cloud trigger that the victim learns to associate with the target class. Rather than imitating the target's overall shape, the trigger is synthesized to elicit the local geometric cues—point density, curvature, and surface structure—on which detectors rely to recognize that class. Because the underlying labels are never modified, the poisoned samples remain label-consistent and pass annotation review, while the trigger is absorbed as a stable signature of the target class. Algorithm 2 states the MIRAGE procedure, and Figure 4.1 illustrates the end-to-end pipeline: surrogate construction, trigger optimization, data poisoning, victim training, and inference-time activation.

Because the study collects no new sensor data, we demonstrate MIRAGE in a scenario built on two open-source datasets, KITTI [24] and nuScenes [28]. The adversary designates car as the target class, and pedestrian and cyclist as the two victim classes. nuScenes supplies public out-of-distribution (POOD) data for pre-training the surrogate. A subset of KITTI target-class data then fine-tunes it; that subset represents the scenes an autonomous vehicle observes while training data is being gathered.

We first construct a surrogate model. Because the victim detector is not directly accessible, we pre-train a proxy on public POOD data to capture general LiDAR features such as ground structure, object contours, and density patterns. This proxy is then fine-tuned on the available target-class samples so that it encodes class-specific signatures. The two-stage design mirrors transfer learning in vision tasks, where broadly trained feature extractors are adapted to more specialized representations.

With the surrogate in place, trigger generation becomes an optimization over the trigger points alone. Given target-class samples $\mathcal{D}_t$ and the frozen surrogate f_{sur}, we seek a patch δ whose insertion drives the surrogate to predict the target class with high confidence at the patch location (Eq. 4.1). Prior clean-label approaches require access to *non-target* samples or rely on cross-class feature separation [38, 39, 114]; ours operates *exclusively on target-class instances*, which substantially reduces the adversary's knowledge assumptions while preserving clean-label validity.

4.2.1 Surrogate Model

To enable reliable gradient-based trigger optimization, we employ 3DSSD [56] as a surrogate for SECOND and PointPillars. Unlike these voxel- and pillar-based detectors, 3DSSD provides a fully differentiable, end-to-end architecture that avoids non-differentiable stages such as hard voxelization, pillarization, and non-maximum suppression (NMS)-based feature formation. This design enables stable gradient propagation directly through raw point coordinates, allowing the trigger to be optimized with respect to genuine geometric sensitivities rather than artifacts introduced by discretization. The victim models differ architecturally, yet all discriminate classes from local geometric cues: curvature, edge continuity [92, 127], height structure, and point-density signatures [26, 94]. 3DSSD's continuous feature processing captures these shared inductive biases.

To further improve gradient quality and to model class-specific geometric structure precisely, we train the surrogate with a single-class prediction head. Multi-class detectors distribute representational capacity across several categories and optimize a competitive softmax objective, which introduces inter-class gradient interference and encourages the model to learn class *boundaries* rather than fine-grained intra-class structure. A single-class formulation instead yields a cleaner "object-vs.-background" decision surface and allows the surrogate to specialize in the geometry of the target class. The result is a smoother optimization landscape, less gradient noise from irrelevant categories, and gradients better aligned with the shape features that characterize the target class.

The surrogate is a 3DSSD model pre-trained on nuScenes and fine-tuned on KITTI car samples with a single-class head. Its weights are frozen throughout trigger optimization, so gradients flow only into the trigger points, never the detector. The primary victim is PointPillars [25], but the same frozen surrogate is intended to transfer to other voxel- and pillar-based detectors. We test this on SECOND [26], a voxel-based detector with a sparse 3D convolutional backbone, as our cross-architecture transfer target (Section 4.3).

In Algorithm 2, $x \oplus \delta$ denotes *point concatenation*, appending the (pose-transformed, oversampled) trigger points δ to the scene cloud x; $\mathrm{clip}_\tau(\cdot)$ rescales a gradient to norm at most τ; and $\odot$ is the elementwise product. Each trigger point carries (x, y, z, ι), where ι is intensity, initialized i.i.d. uniform on $[0.5, 1.0]$ and held fixed throughout: the mask $m_\iota = [1, 1, 1, 0]$ zeroes the intensity gradient, so optimization updates only the point coordinates.

4.2.2 Trigger Optimization

To learn a compact point-cloud trigger, we formulate trigger optimization as an expected-risk minimization problem over the distribution of clean *target-class* samples. The adversary

Algorithm 2: MIRAGE: Clean-Label Backdoor for LiDAR 3DOD

Require: POOD data $\mathcal{D}_{\text{pood}}$; target-class data $\mathcal{D}_t$; clean training set $\mathcal{D}_{\text{train}}$; frozen single-class surrogate f_{sur}; victim $f_{\text{vict}} = \text{POINTPILLARS}$; unified loss $\mathcal{L}$; sphere radius R; train poison rate ρ; per-class deployment budget B; steps I; step size α; gradient-norm bound τ.

Ensure: Optimized trigger δ^*; poisoned training set $\widetilde{\mathcal{D}}_{\text{train}}$.

Phase 1: Surrogate training (offline; frozen at attack time)
Train f_{sur} on $\mathcal{D}_{\text{pood}}$; fine-tune on $\mathcal{D}_t$ {single-class car head}
Phase 2: Trigger optimization
$\delta \leftarrow \delta_0$ {uniform random fill of a sphere of radius R}
for $i = 1$ **to** I **do**
  Sample batch $\{(x, t)\} \subset \mathcal{D}_t$
  $g \leftarrow \nabla_\delta \mathcal{L}(f_{\text{sur}}(x \oplus \delta),\, t)$
  $g \leftarrow \text{clip}_\tau(g)$ {stabilize non-smooth gradients}
  $g \leftarrow g \odot m_\iota$ {freeze intensity gradient}
  $\delta \leftarrow \delta - \alpha\, g$ {trigger update}
end for
$\delta^* \leftarrow \delta$
Phase 3: Clean-label poisoning (training deployment)
Select $\mathcal{S} \subset \mathcal{D}_{\text{train}}$ with $|\mathcal{S}| = \lfloor \rho\, |\mathcal{D}_{\text{train}}| \rfloor$ $\{\rho = 0.005\}$
$\widetilde{\mathcal{D}}_{\text{train}} \leftarrow \{\, x \oplus \delta^* : x \in \mathcal{S} \,\} \cup (\mathcal{D}_{\text{train}} \setminus \mathcal{S})$ {labels preserved}
Phase 4: Victim model poisoning
Fine-tune f_{vict} on $\widetilde{\mathcal{D}}_{\text{train}}$
Phase 5: Trigger activation (inference deployment)
for each non-target object x up to budget B per class **do**
  $x^* \leftarrow x \oplus \delta^*$
  Evaluate $f_{\text{vict}}(x^*)$ {targeted misclassification}
end for

employs a fully differentiable point-based surrogate detector, 3DSSD, and optimizes a small set of learnable trigger points whose insertion into a scene consistently elicits a target-class prediction at the trigger's location. The procedure requires only clean target-class instances, such as car point clouds drawn from public driving datasets. It assumes no privileged access to the victim's annotation pipeline, labels, or model weights. The setting is therefore a realistic one, since such samples are widely available.

Target Proposal Selection. The surrogate emits a dense set of candidate proposals per scene, only a few of which fall near the trigger. Because the attack objective is defined with respect to the trigger location rather than any annotated object, we select proposals through a spatial gating procedure anchored at the patch center. Let $c^* \in \mathbb{R}^2$ denote the bird's-eye view (BEV) projection of the canonical target-box center, and let $\{(\hat{c}_i, \hat{s}_i)\}$ be the predicted BEV centers and scores of the surrogate proposals. We retain only the *local*

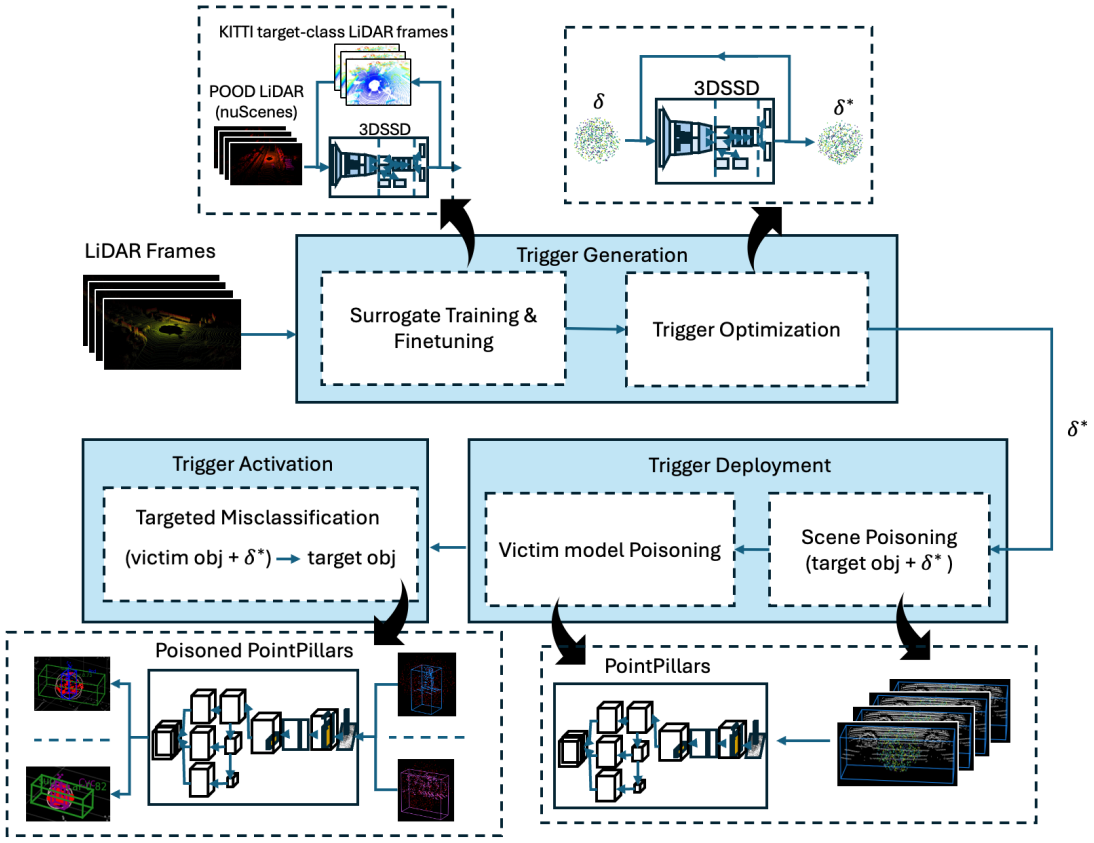


Figure 4.1: MIRAGE pipeline. The adversary pre-trains a surrogate on POOD data, fine-tunes on target-class samples, optimizes a car-inducing trigger, poisons a fraction of target samples, and trains the victim detector on clean + poisoned data. At inference, the trigger induces *targeted misclassification*: non-target classes are detected as the target class.

proposals whose BEV center lies within a fixed radius r_{loc} of the patch,

$$\mathcal{N} = \{\, i : \|\hat{c}_i - c^*\|_2 < r_{\text{loc}} \,\},$$

with $r_{\text{loc}} = 1.5\,\text{m}$ in our experiments; frames for which $\mathcal{N} = \varnothing$ contribute no gradient and are skipped. From $\mathcal{N}$ we form a small working set by taking the union of the K proposals nearest to c^* and the single highest-scoring local proposal ($K = 4$), which retains both well-localized and high-confidence candidates during optimization. The proposal-density term and a coarse working-set center term are evaluated over this working set, whereas the classification term and the finer placement terms are applied to the single proposal nearest the patch center, selected strictly by minimum BEV distance. This provides an unambiguous, distance-grounded optimization target and prevents the trigger from drifting toward spurious distant activations. The canonical target box used to define c^* is *isotropic* ($2R \times 2R \times 2R$),

matching the spherical extent of the patch. We deliberately include no box-size or BEV Intersection over Union (IoU) term. The surrogate head is therefore never biased toward car dimensions, and the trigger's "no car shape" property is preserved. This selection is performed independently at every step, so the optimization target adapts as the trigger evolves.

Formally, let $x \sim \mathcal{D}_{\text{car}}^{\text{KITTI}}$ denote a LiDAR point cloud containing at least one ground-truth car instance, and let $\delta(\phi)$ represent a learnable point-cloud trigger parameterized by ϕ. The trigger is injected via point concatenation, denoted by $x \oplus \delta(\phi)$. We optimize a single unified objective that consolidates the surrogate's detection signal into three groups—classification, placement, and proposal density:

$$\min_{\phi \in \Phi} \; \mathbb{E}_{x \sim \mathcal{D}_{\text{car}}^{\text{KITTI}}} \Big[\mathcal{L}_{\text{cls}}(\tilde{y}) + \mathcal{L}_{\text{loc}}(\tilde{y}, c^*) + \mathcal{L}_{\text{den}}(\tilde{y}, c^*) \Big], \qquad \tilde{y} := f_{\text{sur}}(x \oplus \delta(\phi)) \,. \tag{4.1}$$

The classification term $\mathcal{L}_{\text{cls}}$ drives the confidence of the selected proposal toward one, so the surrogate emits a high-confidence detection at the patch. The placement term $\mathcal{L}_{\text{loc}}$ pulls the predicted box center (in BEV) to the patch center c^*, anchoring the detection to the trigger location. The density term $\mathcal{L}_{\text{den}}$ rewards the presence of multiple surrogate proposals in the immediate neighborhood of the patch, reinforcing a stable, well-supported activation rather than a single fragile proposal.[2] Here ϕ comprises the per-point coordinates together with a per-point intensity channel. The latter is held fixed at its initialization throughout, so ϕ effectively ranges over the trigger coordinates alone. We deliberately omit the vote-concentration and compactness regularizers used in earlier variants: the former has no counterpart in the deployed objective, and spatial extent is instead controlled implicitly, as described next.

Patch Constraints. We impose neither a norm-based constraint on the optimized LiDAR patch nor a per-step projection onto an admissible region. Hard containment of this kind interfered with optimization and degraded trigger effectiveness, so the trigger points evolve freely during synthesis. Spatial extent is instead controlled implicitly through two mechanisms. First, the patch is *initialized* as a uniform random fill of a sphere of radius R (with $R = 1.0\,\text{m}$) centered at the canonical origin, so optimization begins inside a compact volume. Second, we apply gradient clipping at each step (to a maximum gradient norm of 5.0), which bounds the per-iteration displacement of trigger points and stabilizes optimization under the sparse, non-smooth gradients characteristic of LiDAR data. At

[2] In our implementation the three groups expand to seven weighted sub-terms: classification (8.0 on the binary cross-entropy over the proposal probability, 2.0 on the binary cross-entropy over the raw logit); placement (2.0 working-set BEV ℓ_2 distance, 3.0 Smooth-ℓ_1 center alignment on the nearest proposal, 1.0 nearest-proposal BEV distance, 4.0 mean BEV center offset); and density (1.0). We report them grouped for clarity; the coefficients were fixed by preliminary tuning and held constant across runs.

insertion time, the optimized patch is recentered on its own centroid, rotated by the source object's yaw, and translated relative to the object center before being concatenated with the scene points. The trigger is isotropic at initialization and the objective contains no orientation term, yet the yaw rotation is retained as an implicit augmentation. Because the source-object yaw varies from frame to frame, the trigger is exposed to a distribution of orientations during optimization, which encourages robustness to source-object orientation at deployment. The same insertion convention is then applied at deployment, so the trigger is evaluated under the same yaw distribution it was optimized over. Together these choices keep the trigger within the compact, physically realizable patch envelope assumed by our threat model (Section 4.1), while yielding stable optimization and transferable triggers.

Finally, we make the trigger robust to the point subsampling that occurs at both stages of our pipeline. The 3DSSD surrogate selects and groups points via farthest-point sampling and ball query, while the victim detectors retain only a bounded number of points per voxel or pillar; in either case a sparsely injected patch can be thinned before it reaches the detection head. We therefore oversample the patch by a factor of four at insertion, replicating each of the $M = 250$ optimized points to $1{,}000$ deployed points. The replicas are perturbed with millimeter-scale Gaussian jitter so that they are not exact duplicates and can populate neighboring cells rather than collapsing into one. This raises the probability that the trigger's geometry is preserved through both the surrogate's sampling and the victim's voxelization.

4.2.3 Trigger Deployment

After obtaining the optimized trigger δ^*, we inject it into a subset of clean samples to construct poisoned datasets for evaluation. Trigger deployment is separated into two phases corresponding to our experimental objectives: (i) training-time poisoning to assess clean performance preservation, and (ii) inference-time trigger activation to measure targeted attack success. In both cases, deployment modifies only the LiDAR point clouds by concatenating transformed trigger points; all ground-truth labels and annotations remain unchanged. This separation ensures that benign performance (clean precision) and malicious behavior (misclassification / disruption) are evaluated under non-overlapping conditions, preventing metric contamination and isolating the backdoor effect.

Training-Time Poisoning. For clean-precision evaluation, we poison a small fraction of the training frames while preserving all ground-truth labels. Let $\mathcal{D}_{\text{train}}$ be the training split and $\rho \in [0, 1]$ the train poison rate. The poisoned set $\mathcal{S}$ is the contiguous prefix of $\mathcal{D}_{\text{train}}$—under the same frame ordering used for trigger optimization—of size

$$\mathcal{S} \subset \mathcal{D}_{\text{train}}, \qquad |\mathcal{S}| \;=\; \big\lfloor \rho \, |\mathcal{D}_{\text{train}}| \big\rfloor, \tag{4.2}$$

so that $\rho = |\mathcal{S}|/|\mathcal{D}_{\text{train}}|$ is the fraction of training frames that receive the trigger. Each selected frame is poisoned by inserting a single trigger δ^* onto a car instance within it, leaving every ground-truth label unchanged (clean-label); the poisoned training set is

$$\widetilde{\mathcal{D}}_{\text{train}} \;=\; \{\, x \oplus \delta^* : x \in \mathcal{S} \,\} \;\cup\; (\mathcal{D}_{\text{train}} \setminus \mathcal{S}). \tag{4.3}$$

Since $\mathcal{S}$ is a prefix of that ordering, the poisoned frames are a subset of those used for trigger optimization. With $|\mathcal{D}_{\text{train}}| = 3{,}260$ (the car-containing subset of the standard $3{,}712$-frame KITTI training split), the default budget $\rho = 0.005$ poisons $|\mathcal{S}| = \lfloor 0.005 \times 3{,}260 \rfloor = 16$ frames. The victim detector is then fine-tuned on $\widetilde{\mathcal{D}}_{\text{train}}$ from a converged checkpoint to isolate backdoor effects from representation instability. Clean precision after poisoning is compared to a benign baseline to quantify any degradation in standard detection performance. At insertion the patch is ground-snapped, with its lower extent placed on the ground plane (the patch centroid is dropped by half the source object's height and raised by R).

Inference-Time Deployment. To measure targeted misclassification, the optimized trigger is deployed at inference time on non-target classes:

- Pedestrian → Car
- Cyclist → Car

Trigger insertion follows a *frame-level* budget rather than a fractional object rate. For each source class $c \in \{\text{pedestrian}, \text{cyclist}\}$ we patch a fixed number of frames,

$$B_{\texttt{ped}} = B_{\texttt{cyc}} = 100,$$

subject to a *single-patch-per-frame* invariant: each frame receives at most one patch across both source classes. For a frame that contains objects of both source classes, the patch is allocated to the class currently *further* from its remaining budget, with ties broken by a fixed-seed random number generator to keep allocation deterministic. Within a selected frame we choose one source object with bounding box $(x, y, z, d_x, d_y, d_z, \theta)$. The trigger is then transformed relative to that object's pose—ground-snapped and yaw-aligned—and concatenated with the original LiDAR points. No labels or annotations are modified.

We enforce a feasibility condition

$$\text{achieved}_c \;\geq\; 0.95 \cdot \min\big(B_c,\, F_c\big),$$

where F_c is the number of validation frames containing class c, ensuring near-exact budget compliance given the frames actually available.

Evaluation Metrics and Metric Separation. We evaluate two non-overlapping behaviors. Benign behavior is measured on unmodified data as the clean detection mean Average Precision (mAP), together with its change (ΔmAP) relative to a benign baseline. We report KITTI AP_{40}—average precision interpolated over 40 recall points—at the *moderate* difficulty tier, one of KITTI's Easy/Moderate/Hard settings graded by object size, occlusion, and truncation. We refer to this preserved benign performance as clean precision (CP). Malicious behavior is measured only on trigger-activated non-target objects. Because a successful targeted misclassification requires both that the original object disappear and that a car be hallucinated in its place, we report a partition of the suppressed-object population at the detector's confidence score threshold η, evaluated at $\eta \in \{0.1, 0.3, 0.5\}$; we refer to it simply as the score threshold hereafter. The *Misclassification Success Rate* (MSR) counts patches for which the source pedestrian or cyclist is suppressed *and* a car prediction appears within a success radius of $3.0\,\mathrm{m}$ of the patch centroid. Two conditions guard this count: patches and detections are matched one to one, and car predictions overlapping a real car ground-truth box with IoU ≥ 0.1 are excluded, so that genuine nearby cars are never credited to the attack. The *Disappearance Rate* (DR) counts patches for which the source object is suppressed but no car appears. Their sum is the *Total Disruption Rate*, $\mathrm{TDR} = \mathrm{MSR} + \mathrm{DR}$: the fraction of source objects whose detection is disrupted, whether morphed into a car (MSR) or simply suppressed (DR). MSR and DR therefore partition TDR by construction. The IoU thresholds used for the suppression and matching tests are 0.5 for pedestrian and cyclist and 0.7 for car.

Reproducibility. All deployment metadata (per-class frame budgets, source/target classes, random seed, and per-object patch records) are stored in a structured manifest to enable deterministic reconstruction. Clean precision and the disruption metrics are evaluated under strictly separated conditions: CP is computed on unmodified samples, while MSR, DR, and TDR are measured only on trigger-activated non-target objects.

Cross-Model Evaluation. Our primary setting pairs KITTI with a pillar-based PointPillars victim. We additionally evaluate transfer to SECOND, a voxel-based detector using sparse 3D convolution, which tests robustness across architectural inductive biases; because the surrogate is point-based, the study spans all three dominant LiDAR representation families—point (surrogate), pillar (primary victim), and voxel (transfer target).

4.3 Experimental Results

4.3.1 General Setup

Implementation and Hardware. MIRAGE is implemented in PyTorch 2.1.2 (CUDA 11.8) on top of the MMDetection3D 1.4.0 / MMEngine 0.10.4 toolkit (OpenMMLab).

Victim training is distributed across two GPUs, whereas trigger optimization and all evaluation run on a single GPU; all experiments use NVIDIA GeForce RTX 4090 GPUs.

Models. All main experiments use PointPillars [25] as the victim detector; SECOND [26], a voxel-based detector with a sparse 3D convolutional backbone, serves as a cross-architecture transfer target. Because the surrogate is point-based (3DSSD) and the primary victim is pillar-based (PointPillars), evaluating against the voxel-based SECOND tests whether the trigger transfers across all three dominant LiDAR representation families. The trigger is optimized against a frozen 3DSSD [56] surrogate with a single-class (car) detection head. We choose 3DSSD for its full differentiability—no hard voxelization, pillarization, or NMS-based feature formation—so that clean gradients reach the trigger points; Section 4.2 develops the rationale.

We use PointPillars as the primary victim because it exemplifies the pillar-based design that balances accuracy and real-time inference. PointPillars [25] discretizes the point cloud only in the horizontal plane while leaving the vertical axis unbounded: each pillar aggregates a sparse set of points, which are encoded with a lightweight PointNet-style [92] network to learn local geometric features. These pillar features are scattered into a dense 2D pseudo-image, enabling standard 2D convolutional backbones and single-stage detection heads for efficient 3D bounding-box regression and classification. By avoiding 3D convolutions altogether, PointPillars substantially reduces computational overhead while retaining the spatial information necessary for accurate localization. Figure 4.2 provides an overview of this architecture, including the pillar-based encoding, 2D backbone, and detection head.

This design suits large-scale autonomous driving datasets such as KITTI, where fast inference and scalability are critical. PointPillars is a common baseline in 3DOD research, valued for its simplicity, strong performance, and compatibility with frameworks such as `MMDetection3D`. These same properties make it a practical target for security and robustness studies: its modular pipeline permits controlled manipulation of point-level inputs and intermediate representations without altering the detection architecture.

Datasets. nuScenes [28] serves purely as public POOD data for surrogate pre-training. KITTI [24], which provides paired camera RGB images and LiDAR point clouds, is the evaluation benchmark, with car as the target class and pedestrian and cyclist as the two

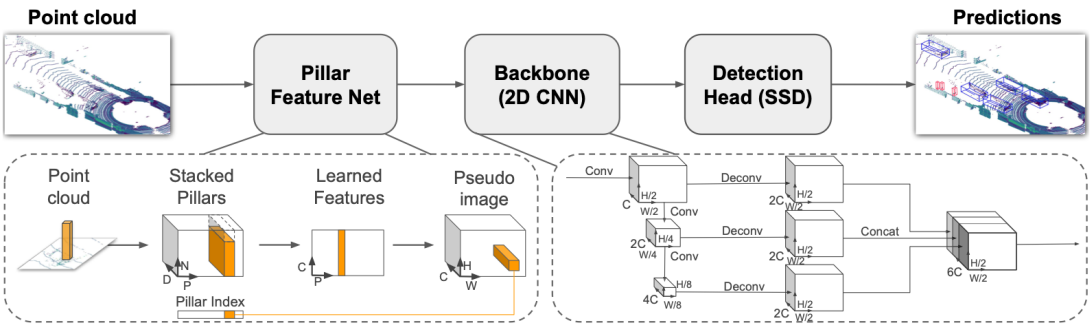


Figure 4.2: Architecture of PointPillars, illustrating pillar-based feature encoding, 2D convolutional backbone, and 3D detection head [25].

victim classes. Figure 4.3 shows an example of each class in the camera view and the LiDAR BEV. We evaluate on KITTI because it is the community-standard, reproducible LiDAR 3DOD benchmark. It places our results on the same footing as prior LiDAR 3DOD backdoor work [33–36]. Injecting the trigger's modeled LiDAR signature into the point cloud also isolates the learning-time vulnerability from the confounds of physical fabrication and field collection. We accordingly frame this study as a controlled, simulation-based proof of feasibility for the weakest realistic attacker of Table 1.1; the physical-fabrication caveat is treated in the threat model (Section 4.1) and as future work in Chapter 5.

The KITTI [24] dataset is one of the most widely used 3DOD benchmarks in autonomous driving, providing synchronized LiDAR point clouds, RGB images, and calibration data from a vehicle-mounted sensor suite in urban and suburban environments. For 3D detection, the dataset contains 7,481 annotated LiDAR frames for training and validation, along with 7,518 test frames whose labels are withheld for online evaluation. Each LiDAR frame is a single driving-scene snapshot and may contain multiple object instances; for example, the car class alone includes 14,357 labeled instances distributed across 3,260 frames in the commonly used train split. In practice, the 7,481 labeled frames are usually split into 3,712 training and 3,769 validation frames, following standard community conventions. KITTI annotations include full 3D bounding boxes with position, dimensions, and orientation for multiple object categories, making KITTI a foundational benchmark for evaluating LiDAR-based 3DOD models under realistic driving conditions.

Surrogate Training. The surrogate is trained with AdamW under a LinearLR warmup (start factor 0.1, 1000 iterations) followed by a MultiStepLR decay ($\gamma = 0.1$), with gradient-norm clipping at 35. POOD pre-training on nuScenes uses learning rate 5×10^{-3}, weight decay 0.01, 40 epochs (milestones $[30, 35]$), $16{,}384$ points per frame aggregated over 10 sweeps, and batch size 12. KITTI car fine-tuning uses learning rate 1×10^{-4}, no weight

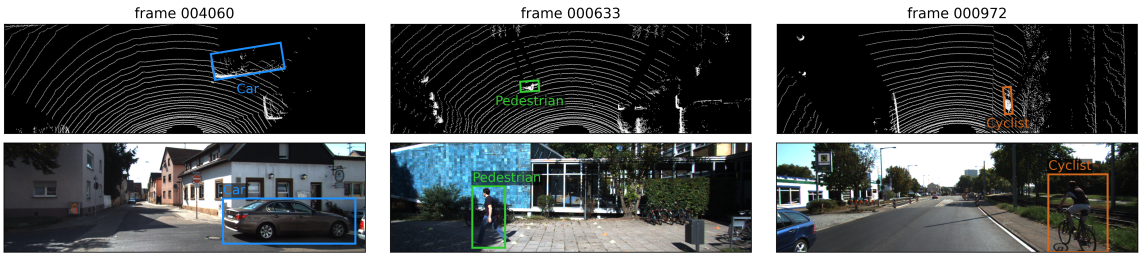


Figure 4.3: Paired top-view point cloud (top) and RGB camera (bottom) images from the KITTI dataset, illustrating the three object classes used in our evaluation: a car (frame 004060, left), a pedestrian (frame 000633, middle), and a cyclist (frame 000972, right). Ground-truth boxes are overlaid in both views. The BEV views highlight a key challenge for LiDAR 3D detection: the number of returns on an object varies sharply with class and distance, with pedestrians and cyclists yielding far sparser clusters than cars.

Table 4.1: Targeted misclassification across victim detectors on KITTI *val* (moderate difficulty). MIRAGE versus a random-sphere trigger of identical geometry, point budget, and deployment protocol. PointPillars is evaluated at $\rho = 0.5\%$, $R = 1.0\,\text{m}$, and $4\times$ deployment density; SECOND at a distinct transfer operating point ($\rho = 10\%$, $R = 1.25\,\text{m}$, $5\times$ density, $M = 300$ trigger points). All results are reported at a score threshold of $\eta = 0.1$. The *Mean* block averages over the two victim classes, the best value per (victim, metric) pair is set in **bold**, and every row derives from a single training seed.

		Clean utility		Pedestrian → Car			Cyclist → Car			Mean		
Victim	Method	mAP↑	ΔmAP	MSR↑	DR	TDR↑	MSR↑	DR	TDR↑	MSR↑	DR	TDR↑
PointPillars [25]	Random sphere	55.9	−7.7	29.0	71.0	100.0	55.0	45.0	100.0	42.0	58.0	100.0
	MIRAGE	**57.5**	**−6.1**	**63.0**	37.0	100.0	**83.0**	15.0	98.0	**73.0**	26.0	99.0
SECOND [26]	Random sphere	52.7	−6.0	**47.0**	38.0	85.0	**66.0**	17.0	83.0	**56.5**	27.5	84.0
	MIRAGE	**53.4**	**−5.3**	45.0	34.0	79.0	50.0	20.0	70.0	47.5	27.0	74.5

decay, 60 epochs (milestones $[40, 50]$), 20,480 points per frame, and batch size 4.

Trigger Optimization. The trigger is optimized with RAdam at a constant learning rate of 3×10^{-3} (the script's step-decay schedule is disabled) for 24 passes over the first 600 car-containing KITTI frames, processed one frame at a time, with gradient-norm clipping at 5.0 and gradient-magnitude early stopping (threshold 5×10^{-3}). The trigger comprises $M = 250$ learnable points initialized as a uniform fill of a sphere of radius $R = 1.0\,\text{m}$; per-point intensity is frozen at initialization so that only the coordinates are optimized, and at deployment the points are oversampled $4\times$ to 1,000 points with millimeter-scale jitter. The unified objective is a fixed-weight sum of classification, placement, and density terms whose coefficients are reported in Section 4.2.

Victim Training under Poisoning. The victim starts from the official MMDetection3D 160-epoch KITTI three-class PointPillars checkpoint. It is fine-tuned on the poisoned set

with learning rate 5×10^{-4}, gradient-norm clipping at 35, and a cosine-annealing learning-rate and momentum schedule, over 20 epochs with a $2\times$ `RepeatDataset` ($\approx$40 effective epochs) as a two-GPU distributed job. Detection classes are {pedestrian, cyclist, car}.

Attack Settings. MIRAGE is strictly clean-label: ground-truth annotations are never modified. Training-time poisoning uses a deliberately low budget of $\rho = 0.5\%$ (0.005) of car frames on the primary victim, PointPillars; the SECOND transfer setting is poisoned at its own, much higher rate ($\rho = 10\%$), as reported in Table 4.1. At inference, the trigger is deployed under a fixed per-class frame budget $B_{\texttt{ped}} = B_{\texttt{cyc}} = 100$, subject to the single-patch-per-frame rule defined in Section 4.2.

Attack Goal and Baseline. The attack goal is targeted *misclassification*: a victim pedestrian or cyclist carrying the trigger is detected as a car. To our knowledge, this makes MIRAGE the first attack to pursue clean-label misclassification in LiDAR 3DOD, so no prior clean-label method exists to compare against. The existing dirty-label LiDAR backdoors [33–35] assume a strictly stronger adversary that edits ground-truth annotations and pursue different objectives (Chapter 2), so a direct success-rate comparison against them would not be controlled. The natural and fair baseline is instead a random (unoptimized) trigger of identical geometry, point budget, and deployment protocol. This random-trigger control isolates the contribution of adversarial trigger optimization from the mere presence of additional LiDAR returns near the target, and all attack metrics are reported against it.

4.3.2 Performance Evaluation

We evaluate MIRAGE on the primary victim (PointPillars) at a single fixed operating point—clean-label poison rate $\rho = 0.5\%$, spherical trigger radius $R = 1.0$ m, $4\times$ deployment density, and score threshold $\eta = 0.1$—and defer sensitivity to each parameter to the ablations below. The metrics (clean precision, MSR, DR, TDR), the 3.0 m success radius, and the per-class IoU thresholds are as defined in Section 4.2. Table 4.1 reports MIRAGE against the random-sphere control defined in Section 4.3, across two victim detectors: PointPillars as the primary victim, and SECOND for cross-architecture transfer.

Preserved benign utility. A clean-label attack is useful only if the poisoned model behaves normally on un-triggered data. At the 0.5% budget, the poisoned PointPillars victim attains a clean overall 3-class 3D AP_{40} of 57.5 at moderate difficulty, a ΔmAP $= -6.1$ change from the benign baseline. Because that baseline is the official, *un-fine-tuned* checkpoint, this gap conflates the brief poisoning fine-tune with the poison itself. A random-trigger control trained under the identical schedule incurs a comparable—indeed slightly larger—clean-AP drop (ΔmAP $= -7.7$, against -6.1 for MIRAGE). The poison-specific cost to standard detection performance is therefore negligible: the optimized poison

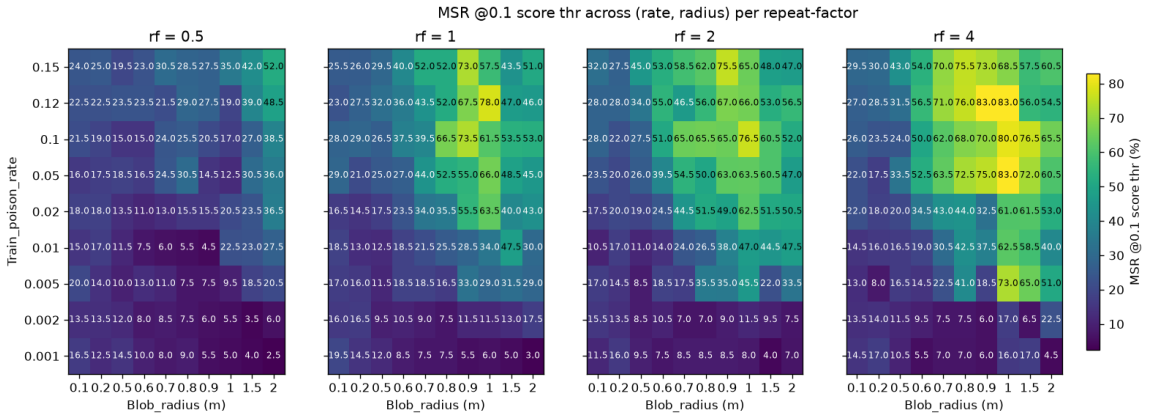
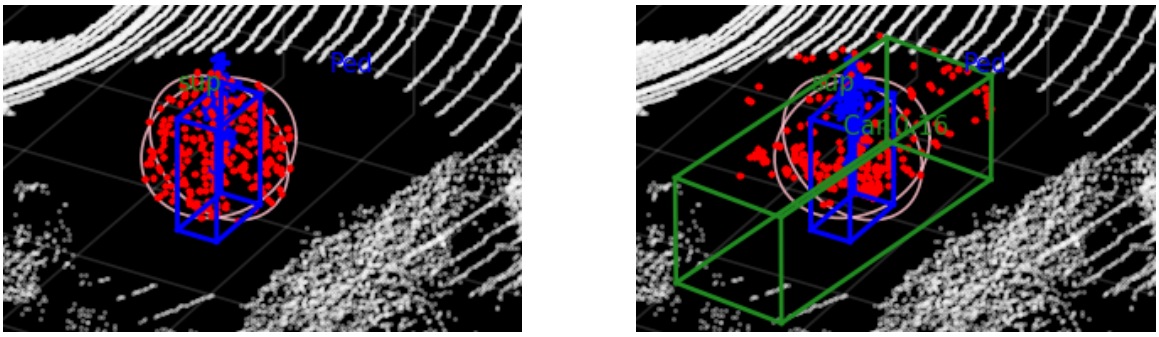


Figure 4.4: Poison-rate $\times$ radius $\times$ density sweep of attack effectiveness on PointPillars. Each cell reports the misclassification success rate (MSR, %)—the fraction of triggered *pedestrian* and *cyclist* instances reported as *car*—at score threshold $\eta = 0.1$, aggregated over both victim classes (*Overall*). Rows vary the clean-label training poison rate ρ and columns vary the spherical-trigger radius R; the four panels fix the deployment-density (point-replication) factor $\mathrm{rf} \in \{0.5, 1, 2, 4\}$, with a color scale shared across all panels.

preserves benign utility at least as well as an unoptimized trigger of identical geometry trained under the same schedule. Because that control also injects trigger points, it bounds the cost attributable to the optimization rather than isolating the fine-tune itself.

Attack effectiveness on the primary victim. On PointPillars the optimized trigger disrupts essentially every triggered source object: total disruption reaches $\mathrm{TDR} = 100.0$ for pedestrian and 98.0 for cyclist at $\eta = 0.1$. Figure 4.5 makes the role of optimization concrete: a random sphere of identical geometry suppresses the source pedestrian but leaves *no car* in its place—a bare disappearance—whereas the optimized trigger replaces it with a co-located *car*. Relative to the random-sphere control, optimization lifts the mean (over *pedestrian* and *cyclist*) misclassification rate from 42.0 to 73.0 while reducing the disappearance rate from 58.0 to 26.0: at a near-ceiling, unchanged level of total disruption ($100.0 \rightarrow 99.0$), it converts bare disappearances into targeted *car* hallucinations rather than raising raw disruption. How that disruption splits between the two failure modes depends on the source class. For cyclist, it is dominated by genuine targeted misclassification—a car is hallucinated at the source location ($\mathrm{MSR} = 83.0$) far more often than the object vanishes ($\mathrm{DR} = 15.0$). For pedestrian, misclassification likewise dominates ($\mathrm{MSR} = 63.0$, $\mathrm{DR} = 37.0$), though the morph to car completes less often than for cyclist, consistent with the sparser return pattern of pedestrians (Figure 4.3) offering less geometric support for a confident car proposal. Because MSR and DR partition TDR by construction, the near-total TDR confirms that the trigger disrupts the source detection even in the frames where it does not finish the conversion—an outcome that, in an autonomous driving context, is itself a



(a) Random sphere (unoptimized control)

(b) Optimized sphere (MIRAGE)

Figure 4.5: A representative *pedestrian* (KITTI *val*, frame 199; close-up 3D view) carrying a sphere trigger of identical geometry, point budget ($M = 250$), and deployment density ($4\times$); the panels differ only in whether the trigger points are randomly initialized or adversarially optimized. Red points are the trigger; the blue box marks the source *pedestrian* ("sup" = suppressed). **(a)** The random sphere suppresses the pedestrian but produces *no car* at the trigger—a bare disappearance. **(b)** The optimized sphere suppresses the pedestrian *and* produces a *car* (green, "Car pred") at the patch—a completed *pedestrian*→*car* misclassification. The aggregate random-vs.-MIRAGE gap is quantified in Table 4.1.

safety-critical failure regardless of whether a phantom car is also produced.

Cross-architecture transfer. Because the trigger is synthesized once against the frozen 3DSSD surrogate and is never exposed to the victim, the same δ^* transfers to detectors with different inductive biases. We evaluate the voxel-based, sparse-convolution SECOND at a distinct transfer operating point ($\rho = 10\%$, $R = 1.25$ m, $5\times$ density, $M = 300$ points), re-tuned for the voxel backbone rather than held fixed. Because the surrogate-to-SECOND transfer places the hallucinated *car* farther from the source than on the pillar-based victim, we score these rows under a relaxed scoring criterion. We accordingly treat them as a transfer assessment rather than a like-for-like comparison. The trigger does transfer, though less successfully than to the primary pillar-based victim: at their respective operating points MIRAGE's mean morph rate falls from $\mathrm{MSR} = 73.0$ on PointPillars to 47.5 on SECOND. This asymmetry follows from how the two victim backbones encode the trigger's points into features. PointPillars collapses the points of each vertical pillar into a single BEV cell through a per-pillar PointNet and max-pool, before any spatial convolution, so a locally dense, orientation-free cluster—exactly the compact blob δ^* forms at the source—is registered largely through its BEV footprint, irrespective of how its returns distribute in height. SECOND instead voxelizes the scene and applies sparse 3D convolutions that encode vertical (z-axis) structure before the feature volume is flattened to BEV, so a trigger that is dense in projection yet carries no coherent car-like 3D structure is partially dampened before it can drive a *car* proposal. Because δ^* is optimized to elicit a confident, co-located *car* from the

surrogate through objectness and ground-plane placement alone—withholding the explicit car-shape prior by design (Section 4.2)—what transfers is precisely this projected-density signature, to which the pillar encoder is maximally exposed and the voxel backbone considerably less so. Both victims ultimately regress boxes from a BEV detection head, so the gap originates in the feature extractor—pillar pooling versus sparse 3D convolution—rather than the output representation; the voxel backbone's weaker spatial selectivity for a structure-free blob is likewise consistent with the farther-displaced *car* responses noted above, which motivate the relaxed success radius we adopt for SECOND. It nonetheless remains effective on the voxel backbone—both the optimized and the random blob disrupt source detections at high rates ($\mathrm{TDR} = 74.5$ and 84.0 respectively, mean over the two victim classes at $\eta = 0.1$)—and here the value of optimization is evident in the *confidence* of the hallucinated *car*, not the raw morph count. At the permissive $\eta = 0.1$ the random sphere even outscores MIRAGE on misclassification ($\mathrm{MSR} = 56.5$ vs. 47.5)—the cells carrying the bold in Table 4.1—: an unstructured dense blob scatters enough returns to elicit many low-confidence *car* proposals near the source. Those proposals are brittle—raising the score threshold to $\eta = 0.5$ collapses the random sphere's morph rate to 8.0 while MIRAGE holds at 25.5, a $\sim 3\times$ gap—so the blob's apparent advantage does not survive a non-trivial confidence requirement. Optimization on the voxel backbone therefore yields *robust, high-confidence* morphs rather than extra raw disruption: random points can suppress an object or elicit a transient *car* proposal, but only the optimized trigger reliably produces one a detector would report.

Poison-rate, radius, and density sweeps. The primary operating point is a single cell of a 360-cell PointPillars grid—9 poison rates ρ spanning 0.1%–15%, 10 trigger radii R spanning 0.1–$2.0\,\mathrm{m}$, and 4 deployment-density (point-replication) factors $\in \{0.5, 1, 2, 4\}$—whose misclassification success rate at $\eta = 0.1$ is summarized in Figure 4.4. Attack effectiveness rises with the poison budget but saturates early: MSR plateaus by $\rho \approx 5$–10% and the strongest cells reach 83 at $R = 1.0\,\mathrm{m}$ (at both $\rho = 5\%$ and $\rho = 12\%$) and at $R = 0.9\,\mathrm{m}$ ($\rho = 12\%$), so we foreground the stealthier 0.5% cell rather than the peak. Along the radius axis the effect is a smooth ramp rather than a discontinuity. At low poison rates MSR stays low for $R \leq 0.9\,\mathrm{m}$ and steps up at $R = 1.0\,\mathrm{m}$, then tapers through $R = 2.0\,\mathrm{m}$. The larger $R \approx 1.5$–$2.0\,\mathrm{m}$ radii lose attack strength without a compensating gain in benign utility, so we retain $R = 1.0\,\mathrm{m}$ as the default. Increasing the replication factor raises MSR with diminishing returns past $2\times$ (median MSR climbs $17 \rightarrow 28 \rightarrow 30 \rightarrow 33$ across $\mathrm{rf} = 0.5 \rightarrow 4$), consistent with better trigger survival through the surrogate's farthest-point sampling and the victim's pillarization, and motivates the $4\times$ default. The two quantities this sweep omits—the disappearance rate and the clean-utility cost—are reported over the same grid in Figures 4.6 and 4.7. These views make explicit how the disruption divides between

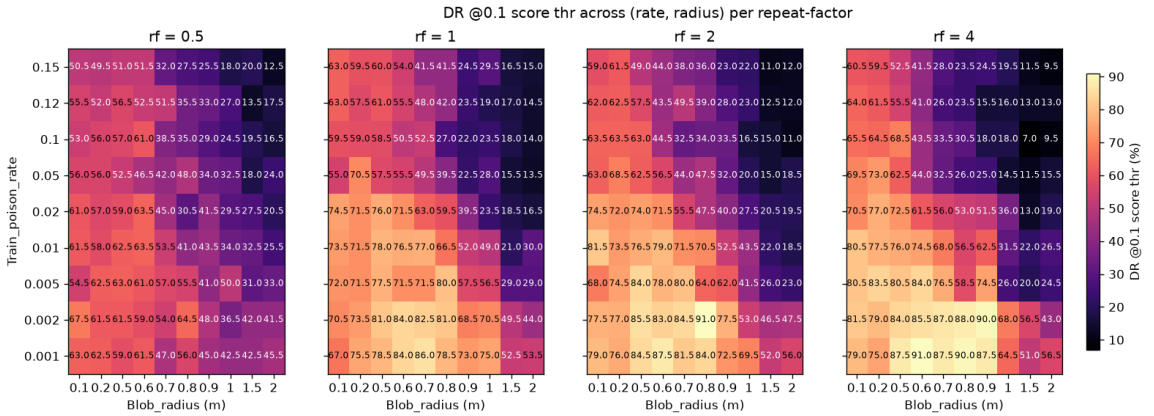


Figure 4.6: Disappearance-rate (DR) companion to Figure 4.4, over the same 360-cell poison-rate $\times$ radius $\times$ density grid on PointPillars (score threshold $\eta = 0.1$, aggregated over *pedestrian* and *cyclist*). Together with the misclassification map, DR completes the total-disruption budget $\mathrm{TDR} = \mathrm{MSR} + \mathrm{DR}$.

the two outcomes. Total disruption $\mathrm{TDR} = \mathrm{MSR} + \mathrm{DR}$ runs near ceiling across most of the grid: the majority of cells exceed 80 even at $\eta = 0.1$. It also varies far less with ρ, R, and rf than MSR does, and is essentially invariant to η, since a higher score threshold shifts mass from MSR into DR. The patch therefore disrupts the original *pedestrian*/*cyclist* detection almost everywhere, and the rate- and radius-dependence of the attack is confined almost entirely to the misclassification channel—whether that disruption resolves into a targeted *car* (MSR) rather than a bare disappearance—with TDR softening only at the largest radii ($R \gtrsim 1.5$ m), where pure disappearances decline faster than misclassifications accrue. In Figure 4.7, the clean-utility cost is flat across the grid: every cell lies between roughly -5 and -9 points of 3-class mAP, with no systematic dependence on poison rate, radius or density. The stealth-first operating point of Table 4.1 therefore gives up no benign utility relative to the aggressive corner; its advantage is the smaller poison budget alone.

The stealth–strength trade-off. Projecting the grid onto the two axes the attacker trades off—attack strength (MSR) and clean-utility cost (ΔmAP)—makes the operating-point choice explicit (Figure 4.8): only 7 of 360 cells are Pareto-non-dominated, and the frontier's strong-attack end is occupied by the dense, $R = 1.0$ m triggers identified above. The front bends sharply near $\mathrm{MSR} \approx 76$: the peak 83 is reached at $\rho = 5\%$, and raising the budget to 10–12% adds nothing. Because clean mAP is flat, the cost of that last stretch is poison budget rather than benign utility—a tenfold larger one than the primary cell's. We therefore operate below this knee, trading a little peak attack strength for a markedly smaller poisoning footprint—the same stealth-first rationale behind the $\rho = 0.5\%$ primary cell.

Trigger-shape ablation. To isolate the role of trigger geometry, we compare the isotropic

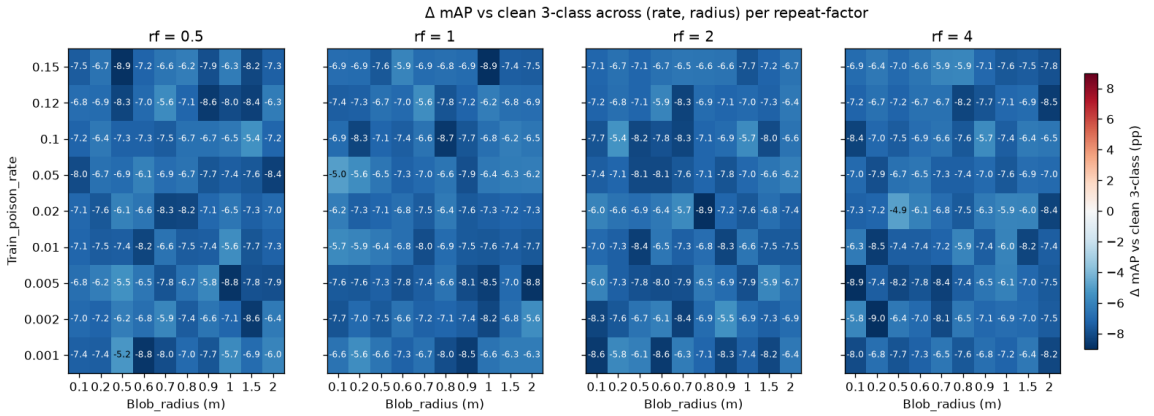


Figure 4.7: Clean-utility cost (ΔmAP) over the same 360-cell grid on PointPillars: the change in clean 3-class 3D AP_{40} (moderate difficulty) relative to the benign baseline. More-negative values (darker) indicate greater benign degradation.

sphere against a car-proportioned anisotropic box with the poisoning budget ($\rho = 2\%$), point budget ($M = 250$), deployment density ($4\times$), surrogate, and optimization schedule all held fixed, so geometry is the only deliberately varied factor. The comparison is a deliberately *confounded* one rather than a clean shape swap, and the confound runs in the box's *favor*: the box variant additionally enables the car-shape priors in the surrogate's target proposal that the sphere intentionally withholds (Section 4.2), so it is given strictly more target-class supervision. Despite that advantage, the box is the weaker trigger. At $\eta = 0.1$ the sphere more than doubles the misclassification rate on both victim classes—MSR $= 66.0$ vs. 29.0 for *cyclist* and 56.0 vs. 23.0 for *pedestrian* (61.0 vs. 26.0 overall)—and lifts total disruption from near-half to near-ceiling (TDR: *cyclist* 98.0 vs. 46.0; overall 97.0 vs. 44.0). The gap is one of *kind* as well as magnitude. The sphere's hallucinated cars are co-located with the trigger—the median patch-centroid-to-*car* distance is 0.0 m, the patch centroid falling inside the predicted box—and it yields *no* phantom cars, that is, no *car* created while the source survives. The box's *car* responses are instead displaced (median 7.7 m), and it yields 27 phantom cars across the 200 triggered victim instances. Figure 4.9 renders a representative *cyclist*: the box (a) leaves the cyclist detected and produces no *car* at the trigger, while the sphere (b) suppresses the cyclist and hallucinates a confident, co-located *car*. We attribute the sphere's superiority to two geometric properties. First, *isotropy*: a sphere presents an identical dense return signature regardless of the source object's heading and the sensor's viewing azimuth, so a single optimized trigger transfers across the wide yaw and range distribution of cyclists and pedestrians. The anisotropic box is orientation-dependent; inserted at a canonical heading, it frequently misaligns with the source geometry and splits or displaces its response. Second, *density concentration*: packing the same point budget into

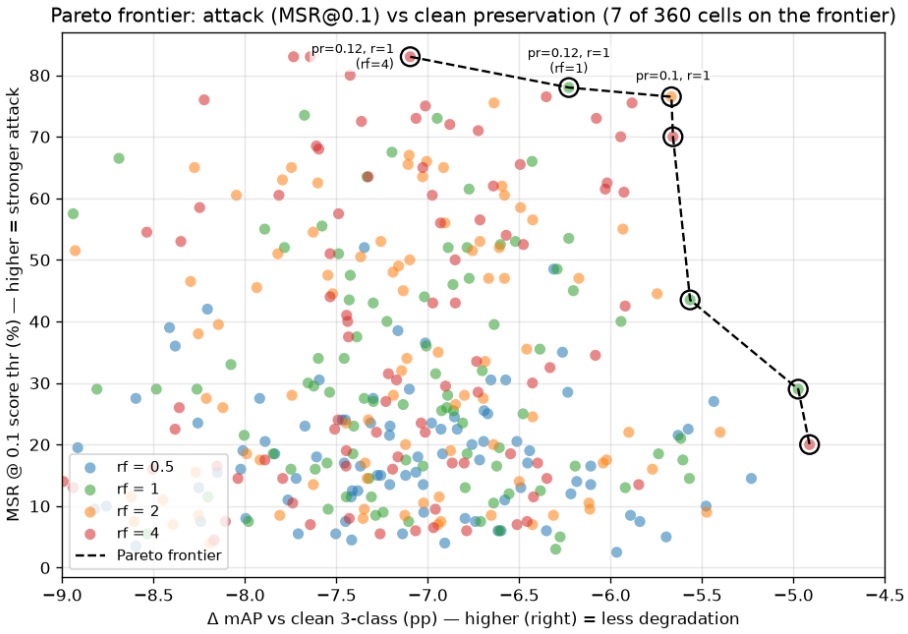


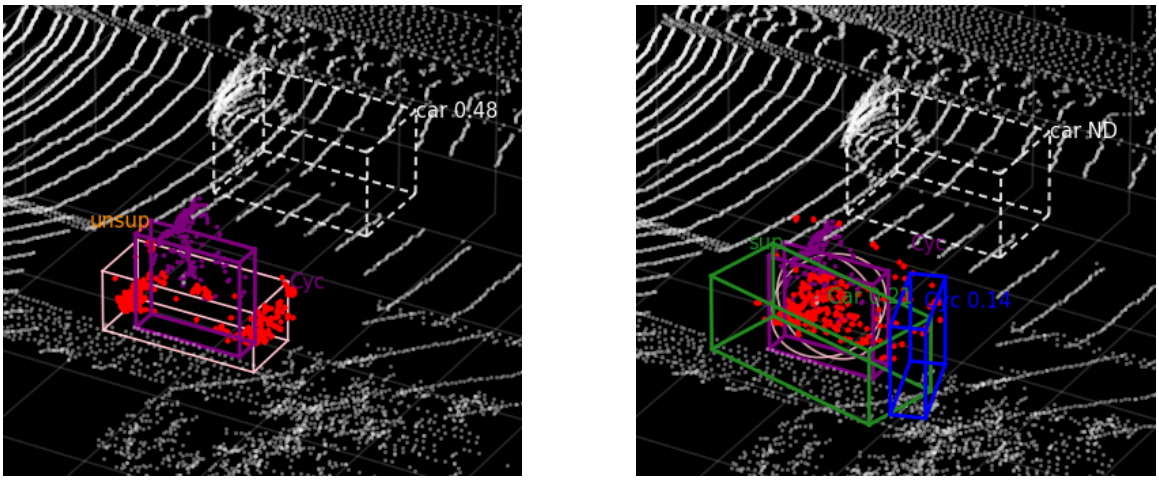
Figure 4.8: Stealth–strength trade-off across the full 360-cell sweep on PointPillars: misclassification success rate (MSR at $\eta = 0.1$; higher = stronger attack) against clean-utility cost (3-class ΔmAP; further right = less degradation). Points are colored by deployment-density factor rf; circled points joined by the dashed line form the Pareto frontier (7 of 360 cells).

a compact sphere yields a far higher *local* return density at the source, saturating the victim's pillar/voxel features and forcing a confident in-place *car* proposal that overrides the true class. The larger, shell-like box spreads the same points over greater extent, weakening the per-cell signal and more often leaving the source intact. The explicit car-shape prior is thus not merely unnecessary but counterproductive: on PointPillars, a dense, orientation-free blob is a stronger trigger than an explicitly car-shaped one.

Robustness to defenses. MIRAGE's trigger is a locally dense cluster of returns, making it a natural target for point-cloud sanitization and backdoor screening. A full empirical defense sweep is beyond the scope of this proof-of-feasibility study; Section 4.5 analyzes the two stages a defender controls—training-time data curation and inference-time input sanitization—and the countermeasures available at each.

4.4 Object Disappearance

The preceding sections develop MIRAGE as a targeted-*misclassification* backdoor: the trigger overwrites a victim's class, so a pedestrian or cyclist is reported as a car. A second safety-critical outcome is *disappearance*: instead of overwriting the victim's class, the trigger suppresses it entirely, so the detector reports no object where one exists. We study



(a) Car-proportioned box trigger

(b) Isotropic spherical trigger (MIRAGE)

Figure 4.9: Trigger geometry on a representative *cyclist* (KITTI *val*, frame 108; close-up 3D view) at the matched ablation setting ($\rho = 2\%$, $M = 250$ points, $4\times$ deployment density). In both panels the red points are the optimized trigger and the white dashed box is a genuine parked *car* elsewhere in the scene (context, not the attack target). **(a)** The car-proportioned box leaves the source detected as a *cyclist* ("unsup"—not suppressed) and produces no *car* at the trigger location. **(b)** The isotropic sphere suppresses the cyclist ("sup") and a confident *car* (green, "Car pred") is hallucinated co-located with the patch—a completed *cyclist→car* misclassification—leaving only a low-score residual *cyclist* box (blue).

disappearance along a single controlled axis—the *objective* used to synthesize the trigger—evaluated across two deployment regimes: training-time clean-label poisoning and pure inference-time evasion. Three trigger objectives are compared. A *misclassification-aligned* trigger reuses the MIRAGE formulation on a cyclist surrogate, so poisoning establishes a phantom-cyclist decoy (Section 4.4.1); a *disappearance-aligned* trigger instead drives the cyclist's own proposals to the background class (Section 4.4.2); and a *dual* trigger combines the two (Section 4.4.3). Section 4.4.4 then examines physical realizability: trigger placement, a deployable pannier-mounted device, and robustness to sensor-realistic ranging noise. Section 4.4.5 carries the study to a second victim architecture, the voxel-based SECOND.

4.4.1 Misclassification-Aligned Trigger

This first attack pursues disappearance through the *same* clean-label, training-time poisoning pipeline developed for MIRAGE (Sections 4.2–4.3). The adversary contributes a small fraction of poisoned training frames whose ground-truth annotations are never altered, and the victim—a KITTI three-class PointPillars detector fine-tuned under the schedule of Section 4.3—absorbs the trigger as a stable signature. Only two components of the MIRAGE procedure change: the surrogate used to synthesize the trigger, and where the trigger is placed

at training versus deployment. The trigger-optimization *objective* is carried over unchanged.

Trigger optimization: shared objective, cyclist surrogate. The phantom trigger is optimized with the same objective equation as MIRAGE, Eq. 4.1: the fixed-weight sum of classification, placement, and density terms, with the same coefficients, gradient clipping, frozen intensity channel, and $4\times$ deployment oversampling detailed in Sections 4.2 and 4.3. The single change is the surrogate: in place of the single-class *car* head, the objective is evaluated on a single-class *cyclist* 3DSSD surrogate fine-tuned on KITTI cyclist data samples. The optimizer is thereby driven to make the surrogate fire a confident *cyclist* detection at the patch, so the synthesized blob is a compact "phantom cyclist"—a dense return cluster that reads, to a cyclist-tuned detector, as a genuine cyclist. Optimization runs on 600 cyclist-containing frames with RAdam at a constant learning rate of 3×10^{-3} (40 steps, gradient-norm clipping at 5.0).

The carried-over objective contains no shape or containment term: Section 4.2 deliberately omits any hard region constraint, and none of the soft containment penalties are applied here. The phantom construction therefore requires one optimization-implementation difference—a *projected-gradient* step that, after every update, re-centers the trigger on its centroid and rescales any point lying beyond radius R back onto the R-ball. Left unconstrained, the classification objective spreads the points well past R (empirically ≈ 1.6 m at $R = 0.3$ m), which would invalidate both the spherical-blob model and the zero-overlap footprint on which the clean-label placement below depends. The per-step projection, with a final backstop clamp, keeps the search inside a radius-R envelope. The blob is initialized as a uniform spherical fill of $M \approx \max(1000\, R^2,\, 20)$ points, which holds areal density roughly constant as R grows while the floor keeps small-R blobs dense. It is deployed at a $4\times$ repeat factor, so the effective on-scene budget is $4M$: 80, 160, and 360 points at $R = 0.1, 0.2, 0.3$ m.

Clean-label poisoning: the placement asymmetry. The disappearance backdoor is created entirely by *where* the phantom is placed: the asymmetry between the training and the deployment position is the whole mechanism. Whereas MIRAGE poisons *target-class* (car) frames with the trigger sitting *on* a car and then activates it *on* a victim pedestrian or cyclist, here we poison *source-class* (cyclist) frames with the phantom placed *beside* the cyclist. Concretely, for a fraction pr of cyclist-containing training frames the phantom is inserted into an empty background cell offset laterally from the rider—ground-snapped, rotated to the cyclist's heading, and guaranteed to have zero bird's-eye-view (BEV) overlap with *every* ground-truth box in the frame. All labels are left untouched, so the poisoning is clean-label: the frame still contains exactly its annotated cyclist, now accompanied by an unlabeled dense blob in its background. At deployment the *same* optimized phantom is instead placed *on* the cyclist, at its BEV center and ground-snapped. By design, this beside-then-on asymmetry

teaches the victim to treat the phantom signature as *background* co-occurring with cyclists during training. At activation the identical signature is dragged onto the cyclist, where it pulls the cyclist's returns toward that learned background association. The detector fails to report the object, and the cyclist disappears.

Evaluation. We fix the deployment placement to *center*—the phantom on the cyclist's BEV centroid—and sweep the blob radius $R \in \{0.1, 0.2, 0.3\}$ m against the poison rate $\mathrm{pr} \in \{0.01, 0.05, 0.10\}$, the fraction of cyclist-containing training frames that are poisoned. Effectiveness is the cyclist *Disappearance Success Rate* (DSR at score threshold $\eta = 0.1$): the fraction of the 100 patched cyclist validation frames in which the detector reports no object at the patch, scored as in Section 4.4.2. Each cell is the mean $\pm$ standard deviation over three full-pipeline seeds (phantom synthesis and victim training share the seed). Because a denser blob suppresses more of the cyclist by sheer occlusion regardless of any poisoning, we pair every poisoned cell with a *clean* baseline: the identical on-cyclist patched validation set scored against the stock 160-epoch PointPillars checkpoint that was *never poisoned*. The *marginal* $\mathrm{DSR}_{\text{poisoned}} - \mathrm{DSR}_{\text{clean}}$ is then the learned lift attributable to the backdoor, above the phantom's raw occlusion floor.

Table 4.2 reports the absolute DSR@0.1 by radius and poison rate against this clean floor. The absolute rate climbs steeply with R—but so does the clean baseline ($44.3 \rightarrow 62.0 \rightarrow 71.0$ with no poisoning at all), confirming that most of the raw disappearance is occlusion. The backdoor's own contribution is the gap above that floor, and it is real for the two larger blobs. At $R = 0.2$ m the poisoned model reaches 73.3 against a 62.0 clean floor, a marginal of $+11.3 \pm 1.5$; $R = 0.3$ m adds $+8.0 \pm 1.0$ over its 71.0 floor. At $R = 0.1$ m the gap stays within noise of zero. The learned effect is strongest at the *lowest* poison rate: $\mathrm{pr} = 0.01$ dominates 0.05 and 0.10 at both larger radii, and with the tightest variance. A budget of roughly one poisoned frame per hundred cyclist frames is thus both the most effective and the most reproducible operating point (bold in Table 4.2). The cost to benign utility is modest and poison-rate-independent (Δ3D-mAP between -6.8 and -7.6, comparable to the brief-fine-tune cost of the MIRAGE poison in Section 4.3).

The contribution of the trigger optimization under center placement. The marginal isolates poisoning from occlusion, but not whether *optimizing* the phantom placed on the cyclist's center—maximizing its cyclist objectness on the surrogate—contributes anything an unoptimized dense blob would not. We test this by running the identical pipeline with a *random* trigger: the uniform-in-sphere initialization deployed with no optimization steps, at the same radius, point budget, and beside-train/on-test placement. It is replicated over the same three full-pipeline seeds as the optimized column, so every entry is a three-seed mean $\pm$ sample std. The comparison is made at $R = 0.2$ m and $\mathrm{pr} = 0.01$, the peak-marginal

Table 4.2: Training-time clean-label poisoning: absolute cyclist Disappearance Success Rate (DSR@η=0.1, %) on a *poisoned* three-class PointPillars victim, by blob radius R and clean-label poison rate pr (fraction of cyclist frames poisoned), with the poison-rate-independent *clean* occlusion floor—the same on-cyclist validation set scored against the never-poisoned 160-epoch checkpoint—in the first column. Deployment placement is *center* (phantom on the cyclist), scored as in Section 4.4.2. Each cell is the mean ± standard deviation over three full-pipeline seeds; 100 patched cyclist validation frames. Bold marks the peak learned marginal, $R = 0.2$ m at pr $= 0.01$ (73.3 vs. the 62.0 clean floor, +11.3).

	Clean floor	Poisoned, by poison rate pr		
R (m)	(stock 160e)	pr $= 0.01$	pr $= 0.05$	pr $= 0.10$
0.1	44.3 ± 1.2	45.7 ± 1.5	46.7 ± 4.5	46.0 ± 3.5
0.2	62.0 ± 2.6	$\mathbf{73.3} \pm 4.0$	71.3 ± 3.1	71.7 ± 4.9
0.3	71.0 ± 1.0	79.0 ± 1.0	78.0 ± 1.0	78.3 ± 1.5

operating point of Table 4.2, whose corresponding cell the optimized column reproduces.

As shown in Table 4.3, the random blob suppresses more than the optimized phantom on every raw measure, and at lower utility cost: it occludes more on the never-poisoned model (66.3 vs. 62.0 clean-floor DSR, a pure-geometry gap), reaches a higher poisoned DSR (78.7 vs. 73.3), and costs *less* benign utility (Δ3D-mAP -6.5 vs. -7.2; $\Delta\text{AP}_{\text{Cyc}}$ -3.3 vs. -4.2). These raw numbers, however, conflate two effects: the sheer occlusion a blob causes and the learned backdoor installed by poisoning. The quantity that separates them is the *learned marginal*—each blob's suppression above its *own* clean floor (the DSR the same blob produces on a never-poisoned model)—which cancels that occlusion and isolates the poisoning's contribution. On the marginal the random and optimized blobs are statistically indistinguishable: $+12.3 \pm 2.1$ for the random blob against $+11.3 \pm 1.5$ for the optimized. The one-point difference favors the random blob but sits well inside the seed spread of either arm. Across three seeds, then, at center deployment the misclassification optimization yields *no* marginal lift over a same-size random blob. The objective *compacts* the blob so that it reads as a single cyclist to the surrogate, which shrinks the occlusion footprint that drives suppression without structuring the points into any stronger a learned trigger.

The same compaction also changes the trigger's *geometry*, and this is where the optimized phantom earns its keep—detectability. Point-cloud input sanitizers are built to strip precisely the signature the random blob presents: a dense, uniform, *volumetric* cluster of returns that fills space rather than lying on a surface. Statistical outlier removal and learned denoisers such as DUP-Net [128] and IF-Defense [129] discard points that sit off the surface or far from their neighbors and restore the rest onto a clean surface, and outlier screens that clear random and adversarial noise are known to weaken specifically on *structured*

Table 4.3: Training-time poisoning at $R = 0.2\,\mathrm{m}$ and $\mathrm{pr} = 0.01$, three full-pipeline seeds: the optimized phantom versus an unoptimized *random* blob of identical radius, point budget, and beside-train/on-test placement (DSR@η=0.1 and utility change, in %; each entry the three-seed mean $\pm$ sample std). The optimized column reproduces the $R = 0.2$, $\mathrm{pr} = 0.01$ cell of Table 4.2. The clean floor is the pure-occlusion DSR on the never-poisoned model, poison-rate-independent (the random blob is deterministic given its seed). The optimized phantom's practical edge is stealth: a uniform volumetric fill is the off-surface, physically-inconsistent signature that point-cloud sanitizers [128, 129] and LiDAR occlusion/shadow detectors [130, 131] are built to remove, whereas its compact, structured return is harder to fingerprint—a defensive trade-off rather than an attack-strength win (Section 4.5).

$R = 0.2\,\mathrm{m}$	Random blob	Optimized
Clean-floor DSR	66.3 ± 1.5	62.0 ± 2.6
Poisoned DSR	78.7 ± 1.2	73.3 ± 4.0
Learned marginal	$+12.3 \pm 2.1$	$+11.3 \pm 1.5$
Δ3D-mAP	-6.5 ± 0.9	-7.2 ± 0.7
$\Delta\mathrm{AP}_{\mathrm{Cyc}}$	-3.3 ± 2.0	-4.2 ± 1.3

triggers [132]. The LiDAR setting adds a physical line of defense: genuine returns obey the sensor's ray geometry—each lies on the first surface a beam meets and casts an occlusion "shadow" behind it—so injected points that break these free-space and shadow invariants are flagged by detectors such as CARLO [130] and Shadow-Catcher [131]. A uniform volumetric blob violates both the surface-manifold premise of the sanitizers and the occlusion physics of real LiDAR, whereas the optimized phantom's compact, structured return is closer to a plausible object; the random blob is therefore the stronger attack but the more *removable* one, so the strongest form of this attack is also the most detectable—a natural starting point for the defenses of Section 4.5. We do not run these detectors here (a quantitative detectability comparison is left to future work, Chapter 5), but the direction follows directly from the defense literature. On the axes we measure, the conclusion is unchanged: a same-size random blob matches or beats the optimized phantom on suppression, benign utility, and the learned marginal, so the attack's power is in the poisoning while the optimization buys stealth rather than strength. The center deployment studied here maximizes the phantom–cyclist overlap and is the strongest but not the most physically realizable configuration; Section 4.4.4 discusses realizability. Section 4.4.5 carries this trigger to the voxel-based SECOND victim, alongside the corresponding cross-architecture study for the disappearance-aligned and dual objectives.

4.4.2 Disappearance-Aligned Trigger: The Occlusion Baseline

Disappearance objective. Whereas the MIRAGE objective (Eq. 4.1) drives the surrogate to *emit* a confident car proposal at the patch, disappearance inverts this goal: it drives every proposal near the target toward the background class, so the object is not detected. We optimize the patch on the same fully differentiable 3DSSD surrogate used throughout this chapter (Section 4.2), here trained with a single-class *cyclist* head. As in the target-proposal selection of Section 4.2, let the surrogate emit proposals with BEV centers $\hat{c}_i \in \mathbb{R}^2$ and objectness logits z_i, with score $\hat{s}_i = \sigma(z_i)$. Because the head is single-class, $\hat{s}_i$ is simply the surrogate's confidence that a cyclist is present at proposal i. Given the target cyclist's box center c_o (its BEV projection) and the proposal-locality radius $r_{\text{loc}} = 1.5\,\text{m}$ of Section 4.2, we form the *matched proposal set*

$$\mathcal{M}_o = \{\, i : \|\hat{c}_i - c_o\|_2 < r_{\text{loc}} \,\}, \tag{4.4}$$

the proposals close enough to the cyclist to constitute a detection of it. The disappearance loss assigns every matched proposal a background target, driving its objectness toward zero:

$$\begin{aligned} \mathcal{L}_{\text{dis}} &= \sum_{i \in \mathcal{M}_o} w_i \,\operatorname{softplus}(z_i) = \sum_{i \in \mathcal{M}_o} w_i \big(-\log(1-\hat{s}_i)\big), \\ w_i &= \operatorname{softmax}_i\big(\beta\, z_i^{(\text{detach})}\big). \end{aligned} \tag{4.5}$$

Each term is the binary cross-entropy of a matched proposal against the background (zero) target, using the identity $-\log(1-\sigma(z)) = \operatorname{softplus}(z)$. The weights w_i are a temperature-β softmax over the detached logits. At the default $\beta = 0$ they are uniform and $\mathcal{L}_{\text{dis}}$ is the mean proposal loss. Setting $\beta > 0$ concentrates the gradient on whichever proposal still most confidently reports the cyclist. Minimizing $\mathcal{L}_{\text{dis}}$ sends $z_i \to -\infty$, hence $\hat{s}_i \to 0$, for every proposal in the cyclist's vicinity, so the surrogate ceases to report the cyclist. Whether that suppression transfers to the victim detector is what the evaluation below measures. If $\mathcal{M}_o$ is empty the cyclist is already suppressed, and the loss is identically zero with no gradient. For pure disappearance the full objective is $\mathcal{L} = \mathcal{L}_{\text{dis}}$, with the car-seating term of MIRAGE disabled and no car surrogate built. The suppression anchor c_o is always the cyclist's own box center, independent of where the physical blob is placed.

Placement and deployment. The blob is initialized as a uniform spherical fill of radius R containing $M \approx 750\,R^2$ points, which holds areal density constant as the radius grows. It is oversampled $4\times$ with millimeter-scale jitter at deployment, following the insertion convention of Section 4.2. We evaluate two placement strategies that share the suppression anchor above but differ in the blob's physical position relative to the cyclist. *Center* places

the blob at the cyclist's BEV position, ground-snapped; *rack* shifts it rearward over the rear wheel and sets its height from the blob size, approximating a rack- or pannier-mounted device. In both cases the blob is rotated to the cyclist's heading. Rack is a geometric heuristic—KITTI carries no rack annotation—and is physically sensible only for the smaller radii.

Evaluation. To isolate the pure test-time component of the attack, we deploy the optimized patches against a clean, standard 160-epoch KITTI three-class PointPillars checkpoint that was *never poisoned*—the same benign baseline used in Section 4.3—and evaluate on the 100 patched cyclist validation frames. For each placement we sweep the blob radius $R \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ m and report the *Disappearance Success Rate* (DSR), the counterpart for this attack of the disappearance rate (DR) of Section 4.2: the fraction of patched cyclists for which the detector reports no object at the patch, at score threshold $\eta = 0.1$. Each cell is reported as the mean $\pm$ standard deviation over three independent optimization seeds. The optimized blob is also compared against a *random-sphere* control—an unoptimized blob of identical radius, point budget, and deployment protocol—which plays the same role here as for MIRAGE (Section 4.3): separating the contribution of the disappearance-loss optimization from the mere injection of a dense cluster of returns.

In Table 4.4, the optimized blob and the random-sphere control are compared across the radius sweep under both placements, every cell a three-seed mean DSR on a detector that was never poisoned. In Figure 4.10, the same four series are plotted against radius, each climbing monotonically as the denser blob corrupts more of the cyclist's pillar features. Reading the random sphere as the *occlusion floor* of an unoptimized dense blob of matched size and density, the gap between each optimized curve and that floor isolates what the disappearance-loss optimization contributes to the physical evasion—and that contribution divides the two placements. Under *center* placement, with the blob sitting directly on the cyclist, the effect is *occlusion* rather than optimization: the random sphere matches or *exceeds* the optimized blob at every radius (88.0 vs. 69.7 at $R = 0.4$ m). Once a dense cluster of returns is on the target, the optimization contributes nothing. Under *rack* placement the ordering reverses: the optimized blob leads at every radius by $+15$ to $+24$ points (73.0 vs. 57.3 at $R = 0.4$ m). Off the cyclist's centroid an unstructured blob occludes fewer of the relevant returns, and the optimization is what steers the suppression back onto the cyclist. Together these establish disappearance as an inference-time physical-evasion attack powered by dense-return occlusion: it demands no training-time access to the victim and—when the blob can be placed on the target—no attack-specific optimization either. This is the essential contrast with MIRAGE, whose misclassification backdoor requires the victim to have *learned* the trigger→*car* association at training time. Here a standard, un-poisoned detector spontaneously fails to report a *cyclist* whose LiDAR signature is corrupted by a dense blob.

Table 4.4: Inference-time physical evasion: cyclist Disappearance Success Rate (DSR, %) on a *clean*, never-poisoned PointPillars victim, by blob radius R and placement, for the *optimized* blob and a *random-sphere* control of identical radius, point budget, and deployment. Each cell is the mean $\pm$ standard deviation over three optimization seeds, at score threshold $\eta = 0.1$. The stronger attack per placement is in **bold**: the random sphere dominates *center* (on-target occlusion needs no optimization), the optimized blob dominates *rack* (optimization offsets the displaced-blob deficit). 100 patched cyclist validation frames.

	Center		Rack	
R (m)	Optimized	Random	Optimized	Random
0.1	43.3 ± 3.4	$\mathbf{45.0} \pm 1.6$	$\mathbf{48.3} \pm 1.7$	28.3 ± 0.9
0.2	52.0 ± 2.2	$\mathbf{63.7} \pm 1.2$	$\mathbf{60.0} \pm 4.3$	36.0 ± 2.2
0.3	55.7 ± 2.9	$\mathbf{78.3} \pm 1.2$	$\mathbf{63.7} \pm 3.8$	48.3 ± 3.1
0.4	69.7 ± 0.5	$\mathbf{88.0} \pm 0.0$	$\mathbf{73.0} \pm 4.2$	57.3 ± 0.5

Training-time poisoning is counterproductive. The inference-time result above invites the converse test: whether adding the MIRAGE training-time poisoning to this *disappearance-aligned* trigger helps. It does not. We poisoned the victim with the optimized blob under the same clean-label pipeline of Section 4.4.1 and re-evaluated. Table 4.5 places the poisoned success rate beside the clean, never-poisoned detector at every radius and both placements. The clean model's DSR is at least the poisoned model's at seven of the eight operating points ($\Delta = -2$ to $+9$). The single exception, center $R = 0.4$ m, is a -2 near-tie within seed noise. Poisoning does not create disappearance; it *subtracts* a few points by teaching the victim a competing "blob $\rightarrow$ cyclist" association that costs it some of the occlusion-driven suppressions. The penalty is larger for the displaced *rack* placement (mean $\Delta \approx +6$) than for *center* (mean $\Delta \approx +3$). This is why the attack is reported as an inference-time evasion. A disappearance-aligned objective shapes no cyclist-like signature for the victim to bind to suppression. Poisoning therefore has nothing to reinforce, unlike the misclassification-aligned trigger of Section 4.4.1, and the only association it installs works against the attack.

4.4.3 Dual-Objective Trigger

The two preceding triggers isolate opposite mechanisms. The misclassification-aligned trigger of Section 4.4.1 establishes a phantom-cyclist decoy that poisoning reinforces (marginal $+11.3$), but its compaction sheds occlusion. The disappearance-aligned trigger of Section 4.4.2 suppresses by occlusion, yet gains nothing from poisoning. A natural synthesis combines them. The *dual* objective augments the disappearance loss of Eq. 4.5 with a misclassification term $\mathcal{L}_{\text{mis}}$ (the MIRAGE decoy objective of Eq. 4.1, evaluated on the

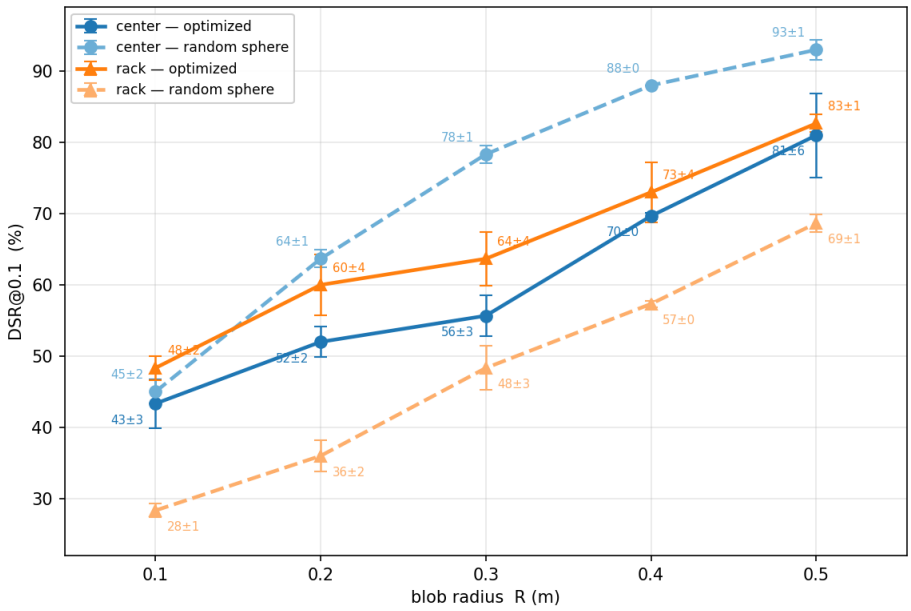


Figure 4.10: Inference-time physical evasion on a clean, never-poisoned PointPillars: cyclist Disappearance Success Rate (DSR at score threshold $\eta = 0.1$) versus blob radius R, for the *optimized* blob (solid) and a *random-sphere* control of matched size and density (dashed), under center (blue) and rack (orange) placement. Markers are the three-seed mean and error bars span $\pm$ one standard deviation. Full values are in Table 4.4.

same cyclist surrogate), so that a single trigger carries both terms at once:

$$\mathcal{L} = \mathcal{L}_{\text{dis}} + \lambda \, \mathcal{L}_{\text{mis}}, \tag{4.6}$$

so the optimizer simultaneously drives the cyclist's own proposals to background and shapes the blob into a confident cyclist decoy at the target center, optimized *on* the cyclist. The victim is then poison-trained under the clean-label beside-then-on protocol of Section 4.4.1.

It is the deployment regime, not the optimization alone, that makes this trigger effective. In Table 4.6, the dual trigger is reported at $R = 0.2\,\text{m}$ and $\text{pr} = 0.01$, both on a never-poisoned detector and after clean-label beside-poisoning. Deployed at inference on the clean detector, the dual blob is outperformed by a plain random sphere exactly as pure disappearance was: its clean floors run 51.3 to 55.3 across $\lambda \in \{0.5, 1, 2\}$, against 63.7 for the random sphere at the same radius and placement (Table 4.4). On target, occlusion still dominates. Poison-training reverses this. The learned marginal over the clean occlusion floor is $+16.3$ to $+18.3$ (three-seed mean, strongest and tightest at $\lambda = 0.5$), exceeding the $+11.3 \pm 1.5$ of the misclassification-aligned trigger (Table 4.3) and reached with *fewer* deployed points (120 versus 160). The lift is trigger-conditioned rather than general degradation: benign cyclist AP falls only -3.9 to -4.4, far below the marginal, and the

Table 4.5: Training-time poisoning is counterproductive for the disappearance-aligned trigger. Cyclist DSR@η=0.1 (%) on a *clean*, never-poisoned PointPillars versus the *poisoned* ($\mathrm{pr} = 0.01$) victim, by blob radius R and placement, both scored as in Table 4.4. Every cell is a single seed, so the clean column differs by a few points from the three-seed means reported there. Δ = clean − poisoned is positive at seven of the eight cells, so poisoning lowers (or at best ties) disappearance; the lone exception (center $R = 0.4$ m, $\Delta = -2$) is a near-tie within seed noise. 100 patched cyclist validation frames.

	Center			Rack		
R (m)	Clean	Poisoned	Δ	Clean	Poisoned	Δ
0.1	42	37	+5	46	37	+9
0.2	50	47	+3	64	60	+4
0.3	59	53	+6	61	54	+7
0.4	70	72	−2	76	72	+4

whole effect comes from roughly five poisoned frames. This is the one configuration in which trigger optimization and poisoning are *jointly* effective: the disappearance term supplies the suppression the misclassification-aligned trigger lacked, while the decoy term supplies the learnable signature that pure disappearance lacked.

In Figure 4.11, the largest absolute suppression in the sweep comes from the two larger radii. At $R = 0.4$ m the dual trigger reaches a cyclist DSR of 90.0 at $\mathrm{pr} = 0.01$ and holds 88.0 at both higher rates; at $R = 0.3$ m the peak is 86.0. The $R = 0.2$ m row peaks at 79.0 and declines to 77.0. The learned marginals do not follow. The clean occlusion floor rises with the radius almost as fast as the poisoned rate does—radius means of 55.7, 60.3 and 67.0 against poisoned means of 78.0, 81.7 and 88.7, a rise of +11.3 against +10.7—so the marginal moves by only −0.7 across the sweep, from +22.3 to +21.7, and no cell leaves the range +18 to +26. Quadrupling the deployed points from 120 to 480 therefore buys occlusion rather than a stronger backdoor, and $R = 0.2$ m is retained: it yields the same learned marginal with a quarter of the points, which is smaller to fabricate and less

Table 4.6: Dual-objective trigger under clean-label beside-poisoning ($R = 0.2$ m, $\mathrm{pr} = 0.01$, center test), by misclassification weight λ. The *marginal* is the poisoned DSR minus the clean occlusion floor on the never-poisoned model. Mean ± sample standard deviation over *three* seeds. 100 patched cyclist validation frames.

λ	Clean floor	Poisoned DSR	Marginal
0.5	55.3 ± 3.5	**73.7** ± 5.5	**+18.3** ± 2.1
1.0	51.3 ± 1.2	68.0 ± 5.3	+16.7 ± 4.6
2.0	51.7 ± 1.5	68.0 ± 5.3	+16.3 ± 5.5

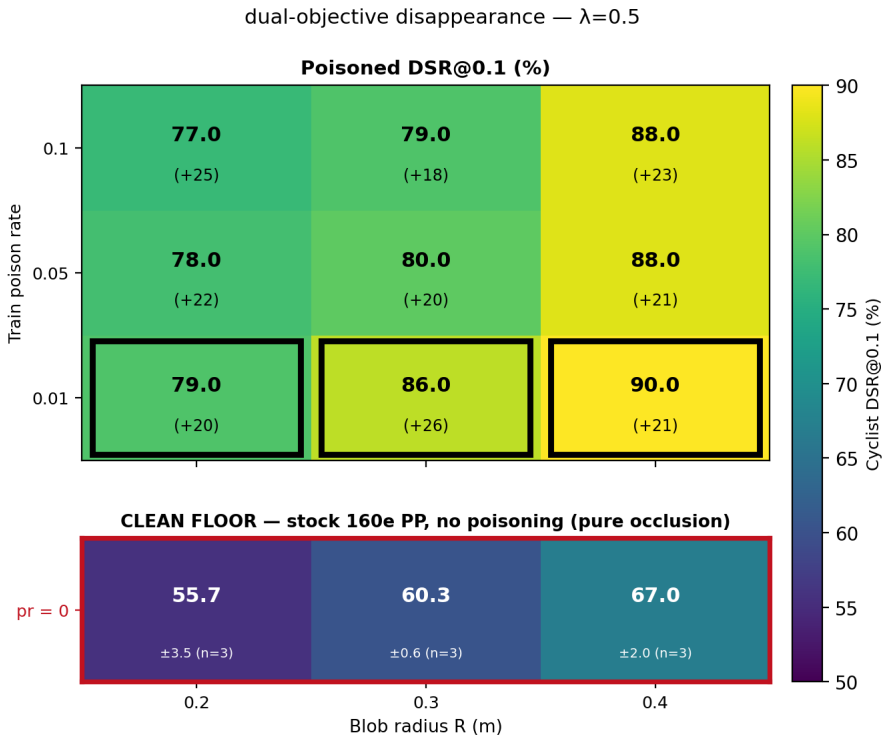


Figure 4.11: Dual-objective trigger at $\lambda = 0.5$: cyclist DSR@η=0.1 over blob radius R and clean-label training poison rate pr, at $4\times$ deployment density on 100 patched cyclist validation objects, scored under the criterion used in Section 4.4. The lower panel is the *clean floor*: the never-poisoned detector on the same patched objects ($\mathrm{pr} = 0$), with each radius reporting the mean over its three separately optimized triggers. In each poisoned cell $(+N)$ is the marginal over that cell's own floor. Black outlines mark the $\mathrm{pr} = 0.01$ row, the operating point of Table 4.6. Every cell is a single seed.

conspicuous at deployment. Benign utility is unaffected by the radius (Δ3D-mAP ≈ -6.2 at all three), so the larger trigger is not paid for in clean performance either. Every cell is a single seed and is read as directional; at the operating point shared with Table 4.6 ($R = 0.2$ m, $\mathrm{pr} = 0.01$) this seed returns a marginal of $+20.0$ against that table's three-seed $+18.3 \pm 2.1$.

4.4.4 Placement and Physical Realizability

The strongest configurations in Sections 4.4.1–4.4.3 place the trigger at the cyclist's center—on the rider—which is not physically realizable. We examine how much of the effect persists as the trigger is brought closer to a deployable form: displaced to a mountable location, replaced by a fabricable device, and subjected to sensor-realistic ranging noise. Each is a small-sample study over two or three seeds, intended to establish the limits of physical realizability rather than definitive success rates.

Trigger placement. With the clean-label pipeline of Section 4.4.1 held fixed, sweeping

Table 4.7: Deployment-placement sweep for the misclassification-aligned trigger ($R = 0.3\,\mathrm{m}$, $\mathrm{pr} = 0.05$, mean $\pm$ sample std over two seeds, DSR@η=0.1). *Clean floor* is the never-poisoned occlusion DSR; *poisoned DSR* is the deployed clean-label attack; the *dirty-label ceiling* relabels the overlapping cyclist to background (an upper bound, neither clean-label nor deployable). Center ($+7.0$ at the same radius and poison rate; Table 4.2) is the non-realizable reference, shown only for comparison.

Deployment placement	Clean floor	Poisoned DSR	Dirty-label ceiling
Centroid	63.0 ± 2.8	71.0 ± 1.4	74.0 ± 0.0
Rear wheel	42.5 ± 2.1	53.5 ± 0.7	55.0 ± 5.7
Torso / carried / in-front	48.5–58.5	43.5–45.5	43.5–50.0

the deployment placement shows the learned marginal to be strongly placement-dependent (Table 4.7). Displacing the blob from the rider's center to the cyclist centroid or over the rear wheel still yields a positive two-seed marginal ($+8$ and $+11$), but every elevated or off-body placement—torso-mounted, hand-carried, or in front of the rider—falls below its own occlusion floor, at a clean-label marginal of -5 to -13. The clean-label effect is thus *placement-locked*: it survives at these two on-body placements and is lost at every elevated or off-body one. Relabeling the overlapping cyclist to background (a dirty-label variant) lifts the ceiling further ($+11$ and $+12.5$), but dirty-labeling is neither clean-label nor deployable and is reported only as an upper bound.

A physically deployable trigger. The floating sphere is an idealization—a ball hovering beside a bicycle. To test a fabricable device we replace it with a *pannier* trigger: two saddlebag-like hemispheres straddling the frame with a $0.30\,\mathrm{m}$ gap, LiDAR returns on the bags only and none on the sub-resolution mount, budget-matched to the sphere and optimized with the same misclassification objective. Table 4.8 compares the two. Optimizing the bags *on* the cyclist rather than beside it more than triples the pannier's clean-label marginal, from $+2.5$ to $+8.5$ over two seeds at center $R = 0.2\,\mathrm{m}$, which confirms genuine learned signal rather than pure occlusion. The deployable geometry still trails the idealized sphere in the mean at every level (best clean-label marginal $+8.5$ vs. $+12.0$; dirty-label ceiling $+11.5$ vs. $+15.0$). Where both are optimized on the object, the pannier's wider two-bag footprint raises its occlusion floor (63.5 against 60.5), leaving less headroom above it.

Robustness to sensor noise. A trigger injected without noise is a more favorable case than one whose returns carry a real sensor's ranging noise. At $R = 0.5\,\mathrm{m}$ under center placement, we raise the deployment jitter from its default $0.003\,\mathrm{m}$ to the accuracy of KITTI's own capture sensor—the Velodyne HDL-64E, $\pm 2\,\mathrm{cm}$ [24]—and re-evaluate over three seeds (Table 4.9). The attack is robust to the added noise: DSR is statistically flat in the jitter

Table 4.8: Deployable pannier trigger versus the idealized floating sphere (center $R = 0.2$ m, $\mathrm{pr} = 0.05$, DSR@η=0.1, mean $\pm$ std over two seeds). *Clean floor* is the never-poisoned occlusion DSR; *poisoned DSR* is the deployed clean-label attack; the *dirty-label ceiling* is the on-cyclist relabel oracle. The clean floor differs by row because the optimization placement changes the deployed patch, so the lift over the floor is the fair comparison. The dirty-label ceiling is an upper bound (neither clean-label nor deployable).

Trigger (opt. placement)	Clean floor	Poisoned DSR	Dirty-label ceiling
Sphere (beside)	65.0 ± 0.0	76.5 ± 3.5	79.5 ± 0.7
Sphere (on-object)	60.5 ± 4.9	72.5 ± 2.1	75.5 ± 6.4
Pannier (beside)	64.5 ± 3.5	67.0 ± 4.2	72.5 ± 2.1
Pannier (on-object)	63.5 ± 0.7	72.0 ± 2.8	75.0 ± 2.8

Table 4.9: Robustness of the disappearance to sensor-realistic ranging noise at $R = 0.5$ m under center placement. Deployment jitter is raised from the default 0.003 m to the KITTI Velodyne HDL-64E accuracy of ± 2 cm (0.02 m) [24]. Mean $\pm$ sample standard deviation over three seeds; DSR@0.3 and @0.5 coincide.

Deployment jitter (m)	DSR@0.1	DSR@0.3/0.5
0.003 (default)	79.0 ± 7.0	80.3 ± 6.7
0.015	86.0 ± 2.6	86.3 ± 2.1
0.02 (± 2 cm)	80.3 ± 2.9	81.3 ± 3.1

magnitude. The sensor-matched $\sigma = 0.02$ m and the default $\sigma = 0.003$ differ by only 1.3 points of DSR@0.1, within the mean cross-seed standard deviation of ± 4.2, and the mean trend is non-monotone, peaking at the intermediate σ. At $\sigma = 0.02$ m the DSR@0.1 is 80.3 ± 2.9, indistinguishable from the 79.0 ± 7.0 at $\sigma = 0.003$, with benign utility unchanged. The disappearance therefore survives the sensor's own ± 2 cm ranging noise across seeds, which makes it physically defensible rather than an artifact of noise-free injection.

Across placement, device geometry, and sensor noise, then, the disappearance holds up in its strongest center forms—inference-time occlusion and the dual backdoor. Clean-label *training-time* disappearance does not robustly survive the move to an elevated or off-body trigger, a limitation we return to in Section 4.6.

4.4.5 Cross-Architecture Transfer: the SECOND Victim

All results above use the pillar-based PointPillars victim. MIRAGE's misclassification backdoor transfers from the point-based 3DSSD surrogate across representation families to the voxel-based SECOND (Table 4.1), so the natural question for disappearance is whether the same holds. It does not: on SECOND the training-time backdoor vanishes entirely, and

Table 4.10: Cross-architecture transfer of the disappearance attack to the voxel-based SECOND victim: absolute cyclist Disappearance Success Rate (DSR@η=0.1, %) by misclassification weight λ, where $\lambda = 0$ is the pure-disappearance objective of Section 4.4.2 and $\lambda > 0$ the dual objective of Section 4.4.3. $R = 0.4$ m ($M = 120$, 480 deployed points), $\mathrm{pr} = 0.05$, clean-label beside-then-on poisoning, center deployment; mean $\pm$ sample std over three full-pipeline seeds; 100 patched cyclist validation frames. *Clean floor* is the never-poisoned SECOND on that variant's own patched validation set—a pure inference-time attack—and *poisoned DSR* the same variant after clean-label poison-training. The *random blob* control skips the optimization loop; it occludes differently and so carries its own floor, and each pair must be read against its own floor, never across the two blocks (one random run per seed is shared across the $\lambda > 0$ rows, the control artifact being λ-independent once optimization is skipped). Bold marks the $\lambda = 0$ optimized clean floor. Each row is scored on the validation objects its own configuration selects, which at $\lambda = 0$ are cyclists only.

	Optimized trigger		Random blob (control)	
λ	Clean floor	Poisoned DSR	Clean floor	Poisoned DSR
0 (pure disap.)	**57.7** $\pm$ 1.5	46.7 $\pm$ 2.9	45.0 $\pm$ 1.7	33.0 $\pm$ 7.0
0.5	48.3 $\pm$ 4.5	37.7 $\pm$ 3.2		
1.0	49.0 $\pm$ 5.3	39.3 $\pm$ 7.4	44.7 $\pm$ 1.5	30.0 $\pm$ 3.6
2.0	47.7 $\pm$ 1.5	43.0 $\pm$ 1.7		

what survives the change of architecture is the *inference-time* half of the attack.

Protocol. We replicate the disappearance study on SECOND [26] at a single operating point, sweeping the one axis that organizes Section 4.4—the trigger-synthesis objective. The dual objective of Eq. 4.6 is swept over $\lambda \in \{0, 0.5, 1, 2\}$, where $\lambda = 0$ deletes the misclassification term outright and recovers the *pure-disappearance* objective of Section 4.4.2 ($\mathcal{L} = \mathcal{L}_{\mathrm{dis}}$), while $\lambda > 0$ is the dual trigger of Section 4.4.3. Everything else is held fixed: the same frozen 3DSSD cyclist surrogate, the same clean-label beside-then-on poisoning protocol of Section 4.4.1, blob radius $R = 0.4$ m ($M = 120$ points at a $4\times$ repeat factor, 480 deployed), poison rate $\mathrm{pr} = 0.05$, center deployment, and the same DSR@η=0.1 scoring criterion used throughout. The victim is a benign KITTI three-class SECOND checkpoint (58.75 AP_{40}), which serves both as the poison-training initialization and, unpoisoned, as the clean occlusion floor on the same patched validation set. Every cell is the mean $\pm$ sample standard deviation over three full-pipeline seeds, on 100 patched cyclist validation frames. Each optimized cell is matched against a *random* blob—identical radius, point budget, placement, poisoned frames, and seed, with the optimization loop skipped—which carries its own clean floor and therefore its own marginal.

The backdoor does not transfer. Table 4.10 reports the absolute rates. On SECOND the learned marginal—poisoned DSR minus that variant's own clean floor—is *negative at*

every λ: -11.0 ± 2.0 at $\lambda = 0$, then -10.7 ± 4.9, -9.7 ± 4.5, and -4.7 ± 2.1 at $\lambda = 0.5, 1, 2$. All twelve optimized cells are negative (mean -9.0, range -14 to -3), and the result is unchanged at score thresholds 0.3 and 0.5. The per-λ error bars overlap, so the only defensible conclusion is that *every* λ is negative; no ordering among them is claimed. This is not general degradation masquerading as an effect: benign utility moves by Δ3D-mAP -5.1 to -6.1 and $\Delta\text{AP}_{\text{Cyc}}$ $+0.4$ to -2.6. The cyclist-AP change is smaller in magnitude than the marginal in every row; the 3D-mAP change is smaller in every row but $\lambda = 2$, where the marginal is weakest at -4.7. The sign of $\Delta\text{AP}_{\text{Cyc}}$ is itself the utility-side shadow of the negative marginal: poison-training leaves SECOND's cyclist AP unchanged or slightly improved, because the poisoned model detects the triggered cyclist *more* reliably than the never-poisoned one. The difference from PointPillars is a sign reversal rather than a change in effect size: on the pillar-based victim the same trigger and the same poison data yield $+15$ to $+22$ (three-seed mean $+17.7$), a gap of roughly 27 points that lies far outside the ± 5 cross-seed spread.

We controlled for the confounds that could produce this sign reversal without any architectural difference. Both victims were trained on the identical poisoned dataset under matched configurations (benign initialization, 20 epochs, and the same optimizer, learning rate, and ground-truth augmentation). Training data, evaluation protocol, and occlusion geometry are matched, and only the *learned* response differs. We attribute the divergence to the victim architecture: an architecture-dependent failure to transfer.

Neither the objective nor the trigger's optimization is at fault. Two controls close off the obvious explanations. Deleting the misclassification term ($\lambda = 0$) changes nothing—its -11.0 ± 2.0 lands mid-band in the $\lambda > 0$ spread and is the tightest row—so the failure is not the decoy objective competing with suppression. Nor is the trigger merely weak: the matched random blob gives -12.0 ± 6.6 at $\lambda = 0$ and -14.7 ± 4.6 at $\lambda > 0$, so skipping optimization does not rescue the backdoor and, if anything, is slightly worse. The differences sit inside the error bars ($\Delta = +1.0 \pm 8.5$ and $+4.0 \pm 9.5$, with the per-seed sign flipping in both arms). Optimized and random are therefore indistinguishable on the learned effect, and both are negative—all eighteen cells across the two conditions.

What does transfer is the inference-time attack. The clean floors tell the opposite story, and they are the most decisive numbers in the table, because a floor is a pure inference-time result: a never-poisoned, unmodified SECOND detector evaluated on patched frames. At $\lambda = 0$ the disappearance-optimized trigger suppresses 57.7 ± 1.5 of the cyclists on that unpoisoned victim—the highest three-seed rate in the SECOND study—against 45.0 ± 1.7 for a random blob of identical radius and point budget. That $+12.7 \pm 3.2$ gap is a *within-*λ controlled comparison (both arms share the same validation frames, source boxes, and placement to within 0.11 mm), and it holds with the same sign on all three

seeds. The surrogate optimization does therefore transfer across representation families—as test-time disruption, not as a backdoor—and poison-training is what destroys it, dragging 57.7 down to 46.7. For the voxel-based victim, then, the deployable form of this attack is inference-time physical evasion: it needs no training-time access, no poisoning budget, and no data-collection compromise, and it is strictly stronger than anything poisoning achieves here. This inverts the PointPillars picture at matched radius: there the random sphere is the stronger inference-time attack under center placement (88.0 vs. 69.7 at $R = 0.4\,\mathrm{m}$, Table 4.4), whereas on SECOND the optimized blob leads by $+12.7$. Which of optimization or raw occlusion dominates the test-time attack is therefore itself architecture-dependent.

Pure disappearance is the better configuration on SECOND. On the clean floor $\lambda = 0$ leads every dual-objective row of Table 4.10, and on the learned marginal neither objective offers anything: poisoning is counterproductive at every λ. The controlled form of that floor comparison, and what it implies about how the objective divides its capacity, are taken up once the third objective has been reported.

The misclassification-aligned trigger fails on SECOND as well. The λ sweep above varies the weight on the decoy term but always retains the disappearance term. The remaining objective of Section 4.4 deletes the suppression term instead: the *misclassification-aligned* phantom of Section 4.4.1, whose objective is surrogate cyclist objectness alone and which contains no disappearance term. On PointPillars this is the trigger that first established a learned backdoor ($+11.3 \pm 1.5$ at $R = 0.2\,\mathrm{m}$, Table 4.3). We replicate it on SECOND at $R = 0.4\,\mathrm{m}$ and $\mathrm{pr} = 0.05$ under the same clean-label beside-then-on protocol and scoring criterion, at a point budget matched to the λ sweep (480 deployed points). A phantom optimized *on* a real cyclist inherits that cyclist's own returns as surrogate objectness, leaving the objective little headroom. We therefore sweep where the phantom is optimized, holding the train-poison and test-deploy pair at a common height so that beside-versus-on is the only difference. Table 4.11 reports the two placements.

The backdoor does not appear at either of them: the marginal is -14.3 ± 3.5 and -17.0 ± 3.6, and the failure is threshold-invariant. The placement sweep matters because it rules out the most obvious objection to a negative result—that the trigger was simply never optimized properly. Only the second placement, which optimizes the phantom *alone* in an empty cell beside the cyclist, produces a genuine descent in the objective (roughly 35% over the first third of the schedule). Its starting loss ranks exactly as saturation predicts—30.2 optimizing alone against 14.1 on the object—because a real cyclist already supplies most of the objectness the phantom is being asked to produce. Fixing the optimization therefore does not rescue the attack: the cell that converged is, if anything, the more negative of the two placements. This is the same conclusion the random-blob controls reach for

the disappearance objectives, arrived at by the opposite route—varying the optimization's *quality* rather than switching it off.

The poisoned victim's failure signature is confident detection rather than weak suppression: across 98 of 100 validation objects the surviving cyclist prediction lies a median of 0.115 m from the patch, so SECOND detects the cyclist precisely where the trigger has been placed. Benign utility again changes by far less than the marginal. Across the two cells Δ3D-mAP is -6.4 ± 1.6 and -6.5 ± 4.5, and $\Delta\mathrm{AP}_{\mathrm{Cyc}}$ is -1.9 ± 2.4 and -2.4 ± 6.5; both cyclist-AP changes are within one cross-seed standard deviation of zero. Neither cell degrades the detector's general performance.

This completes the objective axis on SECOND, and all three objectives yield the same outcome. Whatever the trigger is optimized for—suppression, a decoy, or both—the clean-label beside-then-on poisoning that installs a working backdoor in PointPillars installs none in SECOND, and leaves the victim measurably *better* at the triggered cyclist than the never-poisoned model. We attribute this architecture dependence to the pillar-versus-voxel distinction, an explanation consistent with every result in the SECOND study but directly tested by none of them. PointPillars' pillars are coarse and vertically unbounded, so the beside-placed phantom and the rider fall into overlapping features, and the victim learns a position-conditioned rule—"this blob $\rightarrow$ suppress here"—that the on-cyclist test then activates. SECOND's finer voxel grid keeps the two apart, so the victim learns the narrower and more accurate rule "suppress *the phantom*," which the on-cyclist test does not activate. The cells reported here constrain that explanation in one respect: holding both arms at a common trigger height removes height as a confound. What remains is the roughly 1.06 m of *lateral* separation between the poisoned and activated positions, precisely the axis along which voxel resolution would separate two objects that pillars merge. It remains a hypothesis: testing it would require a sweep over SECOND's voxel size, or a third architecture whose pooling granularity falls between the two.

Optimization yields occlusion only where the objective has headroom—and yields no backdoor either way. To separate the objective from the mere injection of points, each placement was rerun with the optimization loop skipped: a random blob of identical radius, point budget, poisoned frames, placements, and seeds, verified to poison the identical training and validation objects. Each was scored against its own clean floor. Table 4.12 pairs the two cells. On the backdoor the verdict is the same in both: the paired marginal difference is -3.0 ± 6.6 where the optimization converges and 0.0 ± 1.0 where it saturates, the second being an equality to one decimal place. Twelve of twelve individual seeds across the four arms are negative and none is positive, so the phantom backdoor fails on SECOND whether or not its trigger was optimized, and whether or not that optimization worked.

Table 4.11: Misclassification-aligned (phantom) trigger on the voxel-based SECOND victim, by where it is optimized: cyclist DSR@η=0.1 (%) at $R = 0.4$ m, pr $= 0.05$, clean-label beside-then-on poisoning, 100 patched cyclist validation objects. In every row the train-poison and test-deploy placements sit at a common height, so beside-versus-on is the only difference; the rows vary only where the phantom was optimized. *Clean floor* is the never-poisoned SECOND on that cell's own patched objects. Rows are the mean $\pm$ sample standard deviation over three seeds. Bold marks the highest clean floor.

Phantom optimized	Clean floor	Poisoned DSR	Marginal
On the cyclist, box center	50.3 ± 1.2	36.0 ± 3.6	-14.3 ± 3.5
Beside the cyclist (converges)	$\mathbf{54.7} \pm 4.5$	37.7 ± 1.2	-17.0 ± 3.6

On the clean floor the two cells diverge, and this is the sharpest explanatory result in the SECOND study. Where the optimization genuinely descends, it yields $+9.7 \pm 5.5$ points of inference-time occlusion, positive on all three seeds; where it saturates, it yields -2.0 ± 1.0—not merely nothing, but consistently, mildly *worse* than a random blob, negative on all three seeds. The interpretation is that a saturated objective does not stop optimizing so much as stop improving: with no headroom, the forty steps merely drift the patch off its random initialization, and drift is slightly harmful. What makes this more than a restatement of the loss traces is that the two signals come from independent measurement channels: convergence is read off the surrogate during trigger synthesis, the occlusion gain off the victim's clean floor after deployment. They were never fitted to one another, yet they agree in sign on every seed. It also scopes the claim usefully: "optimization yields occlusion" holds for the converging configuration and is simply false at centroid height.

The phantom and pure-disappearance objectives can also be compared directly on occlusion strength. At centroid height the phantom's clean floor is 50.3 ± 1.2 against 57.7 ± 1.5 for the pure-disappearance trigger of Table 4.10, measured on validation objects verified identical seed by seed and at the same point budget. The two ranges do not overlap (49–51 against 56–59). A trigger optimized purely for decoy objectness is therefore the weaker occluder of the two. Taken with the earlier finding that pure disappearance out-occludes the dual objective by $+11$ to $+12$ points, this indicates that occlusion quality tracks how much of the objective is spent on suppression.

Two limits are worth stating. First, the cross-seed spread is wide enough to matter and is itself placement-dependent: the beside cell's marginal ranges from -13 to -20 and its floor from 50 to 59, whereas the centroid cell's floor spans only 49 to 51. Whether any single seed is representative is therefore not knowable in advance: the centroid cell's first seed happened to land on its mean, while the beside cell's did not. This is why single-seed utility

Table 4.12: The contribution of optimizing the phantom. Each placement of Table 4.11 against a matched *random* blob with the optimization loop skipped—identical radius, point budget, poisoned frames, placements and seeds, verified to poison the same training and validation objects. Cyclist DSR@η=0.1 (%) on SECOND, $R = 0.4$ m, $\mathrm{pr} = 0.05$; mean $\pm$ sample std over three seeds; every arm scored against its *own* clean floor, never another arm's. Δ is the optimized-minus-random difference computed per seed and averaged. The two cells differ only in where the phantom was optimized, and hence in whether it had headroom to descend. Bold marks the two clean-floor differences.

	Optimized beside (converges)			Optimized on the cyclist (saturates)		
	Optimized	Random	Δ	Optimized	Random	Δ
Clean floor	$\mathbf{54.7}_{\pm 4.5}$	$45.0_{\pm 1.0}$	$\mathbf{+9.7}_{\pm 5.5}$	$50.3_{\pm 1.2}$	$\mathbf{52.3}_{\pm 0.6}$	$\mathbf{-2.0}_{\pm 1.0}$
Poisoned DSR	$37.7_{\pm 1.2}$	$31.0_{\pm 2.6}$	$+6.7_{\pm 1.5}$	$36.0_{\pm 3.6}$	$38.0_{\pm 5.0}$	$-2.0_{\pm 1.7}$
Learned marginal	$-17.0_{\pm 3.6}$	$-14.0_{\pm 3.5}$	$-3.0_{\pm 6.6}$	$-14.3_{\pm 3.5}$	$-14.3_{\pm 4.5}$	$0.0_{\pm 1.0}$

differences among the placements—including an apparent five-point gain in benign cyclist AP—did not survive replication. Second, the $+4.4$-point floor advantage that the beside placement holds over centroid remains directional rather than established: the two ranges overlap, and separating them would require additional seeds on the noisier of the two cells.

Neither a larger trigger nor a heavier poison turns the backdoor positive. The pure-disappearance objective is the strongest of the three on this victim, so we sweep its two deployment parameters to establish whether the negative marginal is a property of the operating point rather than of the victim. Figure 4.12 reports blob radius $R \in \{0.2, 0.3, 0.4\}$ m against clean-label training poison rate $\mathrm{pr} \in \{0.01, 0.05, 0.10\}$ at $\lambda = 0$, holding the surrogate, the beside-then-on protocol, the deployment density and the scoring criterion fixed. All nine marginals are negative, with a mean of -12.4 ± 3.6 and a range of -20 to -7. Every row and column mean falls inside that spread—-10.7 ± 4.0, -14.7 ± 4.6 and -12.0 ± 1.7 by radius, and -11.0 ± 3.6, -15.3 ± 4.5 and -11.0 ± 1.0 by poison rate—and both axes are non-monotonic, so neither is a lever.

What the radius does move is the occlusion. Quadrupling the deployed points from 120 to 480 lifts the poisoned rate by $+14.0$, from 31.0 to 45.0, and lifts the clean floor by $+15.3$, from 41.7 to 57.0; the two cancel, and the marginal moves by only -1.3 across the sweep. The floor's response is close to linear, rising $+7.7$ at each step. Because the clean floor is measured on the never-poisoned victim, the poison rate cannot reach it, so each radius reports the mean over its three separately optimized triggers rather than a single measurement. Benign utility is unaffected by the radius (Δ3D-mAP -7.9, -6.4 and -6.6 at $R = 0.2$, 0.3 and 0.4 m, and -7.0 ± 1.8 over the sweep), so the larger trigger is not paid for in clean performance; it buys occlusion on both sides of the subtraction.

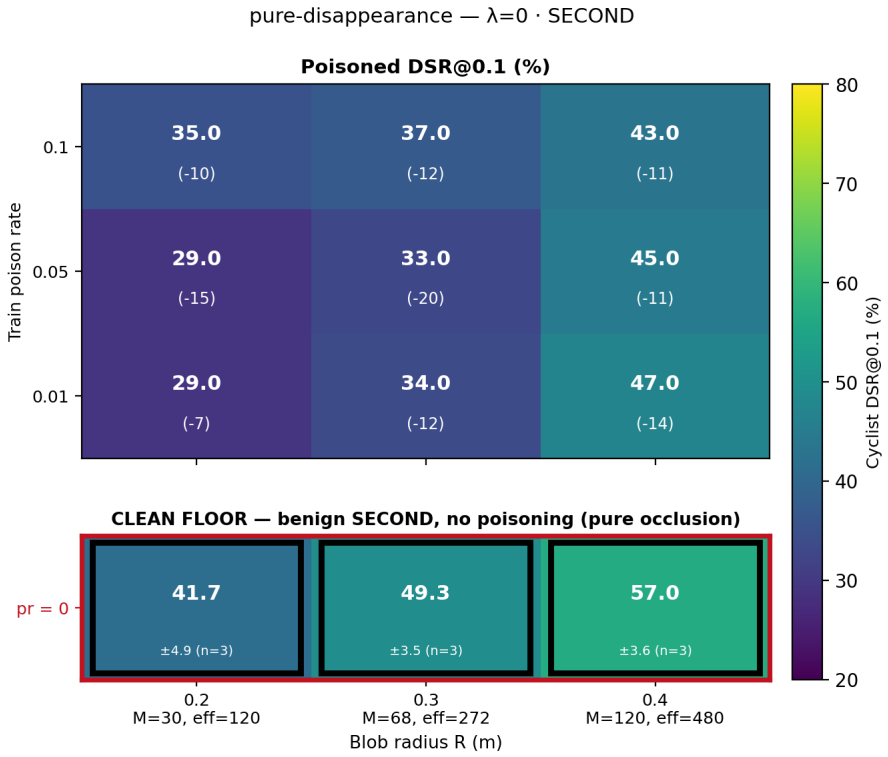


Figure 4.12: Pure-disappearance trigger ($\lambda = 0$) on the voxel-based SECOND victim: cyclist DSR@η=0.1 over blob radius R and clean-label training poison rate pr, at $4\times$ deployment density on 100 patched cyclist validation objects, scored under the criterion used in Section 4.4. The lower panel is the *clean floor*: the never-poisoned SECOND on the same patched objects (pr $= 0$), with each radius reporting the mean $\pm$ sample standard deviation over its three separately optimized triggers. In each poisoned cell $(-N)$ is the marginal over that cell's own floor. Black outlines mark the highest value in each radius column, which is the clean floor in all three. Point budgets are $M = 30$, 68 and 120 at the three radii (120, 272 and 480 points deployed). Every cell is a single seed.

Every cell is a single seed and is read as directional. The least negative cell is $R = 0.2\,\text{m}$ at pr $= 0.01$, at -7.0, which reaches that marginal with a quarter of the deployed points of $R = 0.4\,\text{m}$; it also sits inside the three-seed spread of the $\lambda = 0$ row of Table 4.10 (-11.0 ± 2.0), so no radius or poison rate is claimed to be better than another. Within a single radius, where the poison rate cannot reach the floor at all, the three floors still span up to nine points (36, 44 and 45 at $R = 0.2\,\text{m}$), which is the same order as every difference in the sweep. The floor means reported here average triggers rather than seeds, and are not the same quantity as the floors of Table 4.10: the 57.0 at $R = 0.4\,\text{m}$ averages three separately optimized triggers at one seed, whereas that table's 57.7 ± 1.5 averages one trigger configuration over three seeds. The PointPillars counterpart of this sweep (Figure 4.11) has the same structure and the opposite sign: the marginal is flat there too, neither axis is a

lever, and the radius lifts the poisoned rate and the floor by almost the same amount ($+10.7$ against $+11.3$, for a marginal shift of -0.7), but every cell is positive. The two victims agree on the attack's geometry and its dosage, and differ only in the learned response.

Across the three objectives, the ranking on SECOND is one of occluders, not of backdoors. On the clean floor the pure-disappearance trigger leads. It exceeds every dual-objective row of Table 4.10, and at centroid height it exceeds the misclassification-aligned phantom, 57.7 ± 1.5 against 50.3 ± 1.2 on validation objects verified identical seed by seed. The first of those comparisons is controlled rather than merely observed. Because the generator narrows the poisoned source classes to cyclist alone at $\lambda = 0$, that row is scored on a validation set overlapping the $\lambda > 0$ rows' by only 83 of 100 objects, so we re-scored the $\lambda = 0$ models on the *identical* objects the $\lambda > 0$ rows use, reusing the same checkpoints and the same triggers so that nothing but the evaluation set changes. On matched objects the pure-disappearance floor exceeds every dual-objective floor—by $+11.7 \pm 6.7$, $+11.0 \pm 7.8$ and $+12.3 \pm 4.5$ against $\lambda = 0.5$, 1 and 2 respectively, paired per seed—and the controlled gap is larger than the uncontrolled estimate of $+9.4$, so the frame-set mismatch had been masking part of the effect rather than manufacturing it. The control leaves the backdoor where it was: at $\lambda = 0$ the marginal moves only from -11.0 ± 2.0 to -13.0 ± 3.0. Because a clean floor is a pure inference-time result, pure disappearance is therefore the strongest inference-time physical evasion of the three on this victim.

Two limits attach to that ranking. The pure-versus-dual comparison rests on three seeds, one of which gives a gap of only $+4$. And the lead over the misclassification-aligned trigger is established only at centroid height, where the ranges do not overlap (56–59 against 49–51); against the beside placement, whose floor is 54.7 ± 4.5, the lead is directional and the two ranges overlap. The explanation for the ordering is that the disappearance term $\mathcal{L}_{\text{dis}}$ is what makes the blob an effective occluder, and that adding $\mathcal{L}_{\text{mis}}$ dilutes it. Capacity spent shaping a cyclist-like decoy is capacity not spent disrupting the victim's returns, and it yields nothing on a victim that never learns the decoy. Consistent with this, optimization outperforms the matched random blob by $+12.7 \pm 3.2$ at $\lambda = 0$ (same sign on $3/3$ seeds) but by only $+3.7 \pm 5.1$, with a sign flip, at $\lambda > 0$. Each of those lifts is measured within its own λ, and the random arm was never re-scored on the matched objects, so the contrast *between* them is corroborating rather than controlled.

On the backdoor there is no comparable ranking, because there is no backdoor. The learned marginal is negative for every objective, at every λ, at every radius and poison rate swept, and whether or not the trigger was optimized: across two generators, three objectives, optimized and random triggers, four values of λ, two optimization placements, three blob radii and three poison rates, all thirty-eight cells are negative.

Taken together, the SECOND study sharpens the theme of Section 4.4 rather than complicating it. Training-time poisoning installs a disappearance backdoor only when the victim's representation allows it to bind the trigger to the suppression of a co-located object, and that binding is a property of the architecture as much as of the objective: it holds for pillars and fails for voxels. Inference-time physical evasion carries no such requirement, and it is the form of the attack that survives the change of victim. Testing the voxel-granularity hypothesis directly is left to future work (Chapter 5).

4.5 Defenses

A full empirical evaluation of defenses is beyond the scope of this proof-of-feasibility study; here we analyze where a defender could intervene against MIRAGE and assess the expected efficacy of representative countermeasures. MIRAGE injects its trigger at the data-collection stage, in public space and outside the trust boundary that deployed pipelines implicitly draw around data preparation and model training. The defender—the developer who collects the data, trains the detector, and deploys it—therefore controls only the two stages the attack passes through on its way to the model. Those are *training*, which spans the curation of the collected data and the fitting of the model, and *deployment*, where the trigger is activated at inference. We organize the discussion around these two intervention points and ask, for each, whether the defender can recover the trust signal that the collection-stage compromise removes. Two properties of MIRAGE constrain the analysis throughout: it is clean-label, and it targets detection rather than classification.

4.5.1 Training-Time Data Curation

The defender's first opportunity is to screen the collected data before or during training. The standard poison-detection methods—activation clustering [133], spectral signatures [134], and related latent-space filters—separate poisoned from clean samples by the trace they leave in a model's feature representations. Robust-training and model-cleansing schemes remove the implanted behavior instead: anti-backdoor learning [135] exploits the faster convergence of poisoned examples, and fine-pruning [136] the dormancy of backdoor neurons. Three properties of MIRAGE limit this family. First, because the attack is clean-label, the defender cannot exploit any mismatch between an instance's content and its annotation: the poisoned car instances are labeled exactly as a reviewer would label them and pass the annotation review on which dirty-label pipelines rely. Second, the trigger occupies a compact sub-object region inside a scene that also contains many benign objects, so the poisoned signal is diluted at the granularity these filters operate on—a per-scene or

per-object feature vector aggregates the trigger together with substantial benign structure, weakening the latent separability these defenses assume [137]. Clean-label point-cloud backdoors are already constructed to evade the latent-space and model-inspection defenses they have been evaluated against [52], and MIRAGE inherits this clean-label evasiveness in the harder, scene-level detection regime. Third, and most fundamentally, these methods were developed for classification, where each input maps to one label and one feature vector; a 3DOD model emits a variable set of bounding boxes per frame, and how to define the per-sample statistic these methods cluster or threshold is itself unresolved.

The few backdoor defenses tailored to 3D point clouds inherit this last limitation. CloudFort [132], for instance, mitigates triggers through spatial partitioning and ensemble prediction, but it is built for point-cloud *classifiers* and presumes a single object-level decision to ensemble, an assumption that does not hold for scene-level detection. A more MIRAGE-specific strategy is to search not for a latent anomaly but for the trigger's physical signature directly. The patch design assumes commodity retroreflective sheeting, which saturates the LiDAR intensity channel (Section 4.6.1), so a curator could in principle flag training frames that contain anomalously saturated, spatially compact intensity returns co-located with target-class objects. The same fact that grounds the attack's physical plausibility undermines this defense: intensity-saturated returns already pervade benign data—retroreflective traffic signs produce them throughout KITTI—so a saturation threshold cannot separate the trigger from ordinary roadside infrastructure without an impractical false-positive rate. A defender pursuing this route would need a learned, discriminative model of trigger versus benign saturation, which we leave as an open problem.

4.5.2 Inference-Time Input Sanitization

The defender's second opportunity is at deployment, by purifying each incoming point cloud before it reaches the detector. Because MIRAGE's trigger is a dense, locally structured cluster, the natural candidates are statistical outlier removal and learned denoisers such as DUP-Net [128] and IF-Defense [129], together with geometric and point-reducing input transforms such as random sampling, quantization, and rotation. A recent survey of 3DOD robustness finds geometric transforms among the most effective model-agnostic preprocessing defenses for 3D detectors, with rotation the strongest on average—ahead of flipping and scaling—under non-adaptive perturbation attacks. The same study finds that such transforms confer little robustness once the transform is known to the attacker [29]. These methods are attractive because they are model-agnostic and need no retraining. Three considerations temper the expectation that they neutralize MIRAGE. The first is adaptivity: the trigger geometry is the output of an optimization, and prior work shows that an attacker optimizing

a 3D point cluster can shape its local geometry to evade point-cloud preprocessing and anomaly detectors while preserving its effect [48]. The second is a utility cost specific to this attack's targets. MIRAGE's victim classes—pedestrians and cyclists—are represented by few, sparse returns at range. Aggressive outlier removal or downsampling erodes these returns, so the defender hardens the model against the trigger at the price of degrading detection of the very objects the attack endangers. The third is a mismatch of scale: these methods were designed for single-object clouds of a few thousand points, not full driving scenes of order 10^5 points, and their behavior in the detection setting is largely uncharacterized.

Point-cloud-specific backdoor detectors narrow but do not close this gap. PointCRT [116] detects triggered inputs at inference without prior knowledge of the trigger by measuring robustness to corruption, but it is formulated for classification and requires clean reference samples, and adapting a corruption-robustness signal to the set-valued output of a detector is non-trivial. Generic point-cloud outlier filters, designed to remove random and adversarial noise, have likewise been observed to provide limited efficacy against deliberately placed backdoor triggers [132]. The trigger's status as plausible scene geometry rather than statistical noise is exactly what allows it to survive input purification.

4.5.3 Synthesis

Across both intervention points the same pattern recurs: existing backdoor defenses were built for a different setting—image classification, single-object point clouds, dirty-label poisoning, or a pipeline-insider adversary–and MIRAGE's clean-label, scene-only, detection-targeting construction removes the signal each relies on. We do not claim the attack is undefendable. An effective defense would instead have to be co-designed against this threat model: a curation stage discriminative enough to separate the trigger's saturated signature from benign retroreflective infrastructure, or a runtime purifier that suppresses the trigger without sacrificing the sparse returns of small, distant objects. A third defense family—model inspection by trigger reverse-engineering, such as Neural Cleanse [138]—we do not pursue, because it presumes a small, bounded input space and a per-class trigger to recover, both ill-defined for a detector with set-valued outputs and a physically realized 3D trigger. Quantifying the residual risk MIRAGE poses under adaptive, co-designed defenses is the natural next step, and we identify it as future work (Chapter 5).

4.6 Discussion

4.6.1 Physical Realizability

Physical fabrication and field validation of the patch remain future work; we validate the attack digitally, using effective reflectivities at the sensor that commercially available materials readily achieve and in practice exceed. The patch assumes discrete reflective elements on a scaffold of low LiDAR visibility. Automotive LiDAR occupies the near infrared [35, 139]—905 nm in most units, with 865 and 940 nm variants behaving identically and a 1550 nm premium tier—so a design suited to sensors such as Velodyne and Ouster generalizes across deployed units. Thinner struts are required as the attack range shrinks and the sensor's beam footprint and angular pitch tighten, but the requirement is mild. A strut of 2 mm or less stays below the finest current sensor's sampling even at the closest realistic range ($\sim$5 m), relaxing to $\sim$4 mm at 10 m and $\sim$8 mm beyond 20 m. A single sub-2 mm scaffold is therefore sub-resolution for essentially every sensor in service. A carbon-pile, near-infrared-absorptive coating holds those struts near the noise floor at both 905 and 1550 nm—the Musou Black IR Flock Sheet, for instance, has an effective reflectivity at the sensor of at most 0.5% through 1500 nm. Ordinary visible-black paints such as standard Musou Black or Black 3.0 transmit in the near infrared and do not serve. The requirements on the reflective elements are equally loose. Polished chrome spheres ($\sim$90–95% effective reflectivity at the sensor at 905 nm, $\sim$95–98% at 1550 nm, and pose-isotropic) saturate the intensity channel at every LiDAR wavelength, as does prismatic retroreflective film. Each sensor clips such returns [140]—retroreflective traffic signs already appear as intensity-saturated regions in KITTI—so the elements need only exceed the saturation threshold rather than realize a calibrated value. Sub-2 mm struts, a near-infrared-absorptive black, and chrome reflectors are thus each universal across current automotive LiDAR, so the attack's physical realizability does not hinge on the specific KITTI/Velodyne sensor. The remaining requirements for physical deployment—keeping the trigger co-located with its victim, and the device's modest visual-stealth cost—are kinematic and operational rather than sensor-dependent. Beyond the sensor's material response, the attack also tolerates its *ranging noise*: at $R = 0.5$ m under center placement, raising the deployment jitter to the KITTI-matched ± 2 cm leaves the disappearance essentially unchanged across three seeds (Section 4.4.4).

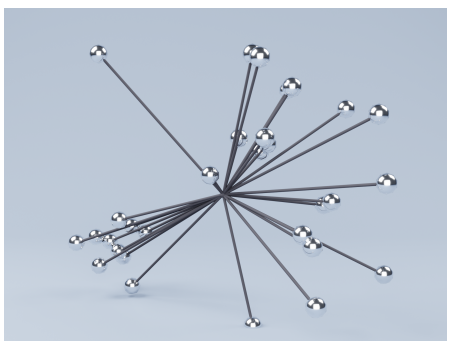
Because MIRAGE overwrites the class of an existing object rather than synthesizing a phantom one, the trigger must stay co-located with its victim throughout a sequence. In our digital injection the patch is placed at the victim's bounding-box center and rotated to the victim's heading, so that it inherits the victim's position and orientation in every frame. A physical deployment would preserve this coupling by construction. Mounting

the trigger directly on the victim object—worn as a backpack by a pedestrian or affixed to the rear rack of a bicycle (Figure 4.13)—would fix it rigidly in the victim's body frame, so that it travels with the victim's velocity across consecutive LiDAR sweeps and presents the same heading angle as the victim at all times. The attacker would therefore need no tracking, pose estimation, or sensor synchronization: the mount alone would reproduce the fixed-center, yaw-matched placement used in our evaluation, keeping the trigger's position, velocity, and orientation locked to the victim's. The orientation component of this placement is not a precondition for the attack. The trigger is a near-isotropic blob whose objective contains no orientation term, so its effect is largely invariant to yaw, and the rotation to the victim's heading in our injection acts as an implicit augmentation rather than a semantic alignment. A body-mounted trigger would reproduce that heading automatically, but because the attack tolerates substantial orientation error, mounting tolerances are loose, and no precise alignment is demanded of the attacker.

The trigger is not an everyday object, but the cost of this conspicuousness is modest. The device—a small cluster of retroreflective elements on a thin, dark scaffold (Figure 4.13)—presents as an abstract or decorative artifact rather than a recognizable attack apparatus, and could plausibly be taken for an art installation or a piece of street furniture by a casual observer. Its visual stealth is thus moderate rather than high. The low footprint of the attack reinforces this. The backdoor is installed at a low poisoning rate, and at deployment the trigger needs to appear on only a small number of objects to be effective—at most one per scene in our evaluation. A sparse scattering of such devices throughout an environment is therefore unlikely to attract scrutiny, even where a dense or overtly artificial deployment would.

4.6.2 Limitations

Our evaluation has three main limitations. First, the attack is validated entirely in the digital domain: we inject the optimized trigger directly into KITTI point clouds and have not yet fabricated the physical patch or demonstrated the backdoor end-to-end against a sensor in the field, so the physical-realizability argument made above remains to be confirmed empirically. Second, cross-architecture transfer is established only for the detector families we study—a 3DSSD surrogate transferring to PointPillars and SECOND victims—and it is not uniform across the two attacks. MIRAGE's targeted misclassification transfers to the voxel-based SECOND (Table 4.1); the clean-label *disappearance* backdoor does not, and remains a PointPillars result, with only its inference-time evasion carrying over (Section 4.4.5). We accordingly make no claim about detectors with substantially different representations, such as transformer- or range-image-based architectures. Third, MIRAGE optimizes the trigger's geometry alone: the per-point intensity channel is fixed at its initial-



(a) Standalone trigger



(b) Target cyclist



(c) Trigger on the rear rack



(d) Trigger on centroid

Figure 4.13: The disappearance trigger as a buildable object. (a) The optimized $R = 0.2\,\mathrm{m}$ trigger rendered as a physical device: $M = 30$ mirror-finish chrome reflectors on a thin, low-visibility strut scaffold, of overall extent $\approx 0.4\,\mathrm{m}$. (b) The target cyclist (KITTI frame 004949). (c) and (d) The same device mounted on the rear rack and worn on the rider's hip; a body-mounted trigger of this kind stays locked to the victim's body frame and reproduces the fixed-center, yaw-matched placement used in our digital injection. In (c) and (d) the cyclist is rendered at reduced opacity so the trigger is clearly visible; reflectors are shown at $2\,\mathrm{cm}$ diameter in (a) and enlarged to $4\,\mathrm{cm}$ in (c) and (d).

ization by design, so the joint optimization of geometry and reflectance is outside the scope of this work. We regard this last choice as a scoping decision rather than a fundamental constraint—freezing intensity isolates the geometric mechanism we set out to characterize and keeps the trigger's reflectance within the readily realizable range discussed above—but it does leave a jointly optimized, potentially stronger trigger unexplored.

4.6.3 Future Work

Several extensions follow from these limitations. The most immediate is a physical realization of the attack: fabricating the patch from the retroreflective and absorptive materials characterized above and validating the backdoor against a live LiDAR sensor, closing the gap between our simulated injection and a deployable trigger; physically realizable adversarial

objects against LiDAR detectors have been studied for test-time evasion [141], but not for a training-time backdoor. A second direction is to lift the geometry-only restriction and optimize the per-point intensity channel jointly with the trigger coordinates, which may yield a more effective or compact trigger and would let the attacker trade geometric conspicuousness against reflective signature. Third, a surrogate-variation transferability study—varying the surrogate detector and its training data—would map how trigger effectiveness depends on the attacker's choice of surrogate, sharpening the black-box threat model. Finally, extending the evaluation beyond the pillar-, voxel-, and point-based victims studied here to architecturally distinct detectors, including transformer- and range-image-based models, would test the generality of the clean-label mechanism.

4.7 Conclusion

MIRAGE is a proof-of-feasibility demonstration of what is, to our knowledge, the first scene-only, clean-label, model-agnostic backdoor attack against LiDAR 3DOD. Across detector families the attack is strongest on the pillar-based PointPillars victim. It transfers to the voxel-based SECOND at a distinct operating point, where its advantage over an unoptimized trigger of the same geometry is the confidence of the induced detection rather than the raw rate. Unlike the prior LiDAR 3DOD backdoors [33–37], the attacker is neither a malicious data provider nor an insider with system access or knowledge. MIRAGE cannot submit samples, cannot manipulate data annotations, and cannot access the model or training algorithm. Its only capability is placement of a physical object in a public training environment, on or near a victim object that a data-collection vehicle drives past. To our knowledge this is the weakest attacker assumed by any published backdoor attack on LiDAR 3DOD, and the first realization in the 3D perception domain of this *scene-only* threat model, also termed *environmental poisoning*.

CHAPTER 5

Unifying Discussions and Future Directions

5.1 Unifying Perspective

This dissertation studies what it takes to trust a machine learning model that is trained not in a controlled data center but at fleet scale, on data streamed from many vehicles and edge devices that no single party owns or curates. Autonomous driving is the setting where this question matters most: modern perception models are trained on point clouds and images aggregated from large fleets of data-collection vehicles, released benchmarks, and crowd-sourced logs. Two requirements must hold before such a model can be deployed in a safety-critical loop. It must be learnable *reliably* when the training data are distributed across heterogeneous, intermittently connected sources, and it must remain *secure* when some of those sources cannot be trusted. Reliability and security are the two faces of a single shift—away from centrally curated data and toward training data pooled from many partially trusted participants, none of whom the model's owner controls.

The two technical contributions of this dissertation address these requirements in turn. The first, Random Walk Stochastic ADMM (RWSADMM) of Chapter 3, asks whether federated learning can remain reliable under the extreme heterogeneity and opportunistic connectivity that characterize decentralized training, and answers it with a personalized, mobile-server optimization scheme backed by convergence guarantees. The second, the clean-label backdoor attack MIRAGE of Chapter 4, asks whether the perception models that such open pipelines produce can be trusted. It shows that a black-box, clean-label adversary can implant targeted failures—misclassifying chosen objects as another class or making them disappear from the detector's output—into a LiDAR-based 3D object detector by poisoning only raw point clouds.

What links the two is the data pipeline itself. The same absence of a trusted central curator that motivates a federated, mobile-server training scheme is precisely what leaves

a fleet-trained perception model open to training-time manipulation: if no one vets every scene that enters the training set, an adversary can introduce a physically realizable trigger into it. Reliability confronts the *benign* corruption of that shared training distribution by statistical heterogeneity; security confronts its *adversarial* corruption by a hidden trigger. Studying them together, rather than in isolation, is what this dissertation argues is necessary for perception systems built on data that no one fully controls.

5.2 From Reliable Decentralized Training to a Trusted Perception Model

The first part of the dissertation establishes that models can be learned reliably from distributed, heterogeneous, and loosely connected data. RWSADMM replaces the fixed central server of classical federated learning with a movable one that visits clients along a random walk over a dynamic reachability graph, aggregating updates opportunistically as it moves. Personalization is realized through hard inequality constraints among neighboring clients rather than global consensus, an approach that avoids the model bias that consensus-style aggregation introduces under highly non-independent and identically distributed (non-IID) data, and a stochastic subproblem solver keeps the per-client computational burden low. The result is a method that addresses statistical heterogeneity and communication–computation constraints simultaneously, without assuming a persistently reachable server, and that provably converges to a stationary point under mild assumptions.

RWSADMM is a general algorithm, and Chapter 3 validates it as such on standard benchmarks, a proof of concept for reliable decentralized optimization. Its operating assumptions, however, describe the autonomous driving data pipeline almost exactly: a population of vehicles, each collecting data under different conditions and routes, connected to the training process only intermittently. A vehicle fleet is thus a canonical instance of the infrastructure-less, heterogeneous setting the method is designed for, and reliable federated optimization is what allows a shared perception model to be trained across it in the first place.

Reliability, however, only guarantees that the model is learned well; it says nothing about whether the data it was learned from can be trusted. The very openness that makes federated and crowd-sourced training attractive—many contributors, no central gatekeeper—also removes the vetting that would catch a malicious contribution. This is the seam at which the dissertation turns from reliability to security.

5.3 Security as the Other Face of Decentralized Data

Chapter 4 examines the security of LiDAR-based 3D object detection under a threat model that matches the exposure surface of exactly this kind of pipeline. The attacker is black-box, treating the deployed detector as opaque, and clean-label, leaving ground-truth annotations intact so that poisoned scenes survive routine data curation. Operating under both constraints at once, MIRAGE optimizes a compact, physically realizable point-cloud trigger on a surrogate detector and transfers it to unseen pillar- and voxel-based victims, causing the trained model to misclassify pedestrians and cyclists as cars. An adapted trigger objective instead makes those objects disappear from the detector's output, an outcome established on the pillar-based victim. Because the trigger enters the training set through scenes of the sort routinely captured by data-collection vehicles, the attack requires no privileged access—only the ability to contribute data, which the open pipeline grants by design.

Read alongside the first part, this result completes the arc. The same decentralization that reliability engineering treats as a benign statistical obstacle is, from the adversary's vantage point, an opportunity: partial, uncontrolled contributions are the norm, and a model trained on them inherits whatever those contributions teach it. The dissertation's two contributions are therefore not two unrelated systems but two aspects of one condition—learning from data pooled across a fleet—under benign heterogeneity in the first case and deliberate poisoning in the second, with the same pooled training data beneath both.

5.4 Implications for Autonomous Driving Safety

The clean-label backdoor attack MIRAGE, developed in Chapter 4, bears directly on the safety of autonomous driving systems. The failure it induces is the one perception is deployed to prevent: an object present in the scene is reported as something else, or not reported at all, and that error propagates into downstream planning and control.

5.4.1 Vulnerability of Current Pipelines

Current autonomous driving perception pipelines are exposed to clean-label backdoor attacks because their training data are aggregated from many sources, reused across tasks, and shared through public benchmarks and pre-trained checkpoints, so no single party can vouch for the provenance of every scene. Chapter 4 shows what that exposure costs: poisoning 0.5% of training frames installs a persistent targeted backdoor on PointPillars at minimal additional cost to clean detection performance relative to an unoptimized trigger of matched geometry trained and evaluated under the same protocol.

An attack optimized on a surrogate model can generalize to distinct victim architectures, as Chapter 4 demonstrates for targeted misclassification across pillar- and voxel-based victims. That generalization is attack-dependent rather than automatic—the clean-label disappearance backdoor studied alongside it does not transfer to the voxel-based victim—so a defender's choice of architecture can block a particular attack. The protection this affords is narrow, however: it is specific to the attack rather than guaranteed in advance, and it does not extend to the inference-time form of the attack, which needs no poisoning and therefore no learned transfer at all. Robustness accordingly cannot be ensured through architectural diversity or dataset scale alone, and must be argued attack by attack.

5.4.2 Need for Security-Aware Training

Standard validation protocols, which report aggregate accuracy on held-out data, cannot detect a clean-label backdoor that preserves benign performance by construction: in Chapter 4 the poisoned detector's clean detection accuracy differs from that of a control trained under the identical schedule by less than the fine-tuning itself costs. Clean detection performance is therefore not an indicator of model integrity when the malicious behavior is conditionally activated and dormant the rest of the time.

Security-aware training should therefore extend beyond performance optimization to include threat modeling, adversarial robustness evaluation, and data integrity checks: auditing training data for anomalous geometric patterns, stress-testing models under targeted trigger activation, and incorporating defenses that limit the influence of small localized perturbations. More broadly, these results motivate perception pipelines evaluated not only for accuracy and efficiency but also for resilience against realistic and stealthy adversarial manipulation.

5.5 Future Directions

The findings of this dissertation open several avenues for future research.

Reliable Learning at Perception Scale. RWSADMM is demonstrated on standard benchmarks; a natural next step is to carry its reliability guarantees to large perception models trained directly on driving data across a fleet, where model size, communication cost, and straggler effects interact with the heterogeneity the method already addresses.

Federated Multimodal Perception. Autonomous perception is inherently multimodal, combining LiDAR and camera streams. Extending reliable federated training to multimodal, bird's-eye-view fusion architectures—while accounting for modality imbalance, missing

modalities, and the communication cost of large fusion networks—is a promising and largely open direction, and one that would apply the reliability techniques of this dissertation to the same perception systems that its security analysis examines.

Backdoor Detection and Mitigation in 3D Perception. This dissertation focuses on attack construction; the complementary problem is defense. Chapter 4 finds that the standard families—latent-space filters, model cleansing, and inference-time purification—each lose the signal they rely on when the attack is clean-label, scene-level, and aimed at a detector, so the open problem is to co-design against that threat model rather than to transfer existing defenses to it. Two targets follow directly: a curation stage able to separate a trigger's saturated intensity signature from the benign retroreflective infrastructure that pervades driving data, and a runtime purifier that suppresses a locally dense trigger without eroding the sparse returns of the small, distant objects such attacks endanger. At fleet scale both would have to compose with robust aggregation that suppresses malicious contributions without sacrificing personalization, for LiDAR-based and eventually multimodal detectors.

Privacy–Security Trade-offs. Federated training limits visibility into local data, which protects privacy, but also conceals poisoning. Formalizing the trade-off between the confidentiality that decentralization provides and the vulnerability it introduces would help design protocols that balance the two in safety-critical settings.

End-to-End Evaluation Frameworks. Reliability and security are still typically measured apart. Benchmarks that jointly evaluate federated optimization quality and adversarial robustness for fleet-trained perception would make the coupling this dissertation identifies measurable and accelerate progress toward deployable systems.

5.6 Conclusion

Taken together, the two parts of this dissertation trace a single arc—from learning reliably on decentralized data to securing the models it yields. What the second part adds to the first is evidence that the coupling between them is not merely conceptual. An adversary who can place one physical object in a scene that a data-collection vehicle drives past, with no access to the model, the training algorithm, or the annotations, can implant a targeted failure into the detector that scene helps train. Treating reliability and security together, rather than as separate concerns, is, this dissertation argues, a prerequisite for depending on perception systems built this way, and for trusting what they report.

As autonomous platforms become more collaborative and more data-intensive, the questions this dissertation raises—how to train reliably across a fleet, and how to trust what that training produces—will only grow more central to their safe deployment.

APPENDICES

APPENDIX A

RWSADMM Convergence Analysis

A.1 RWSADMM: Convergence Analysis

Our proof of convergence for the proposed stochastic ADMM-based federated learning algorithm is non-trivial. It introduces novel techniques to address the challenge of integrating the mobilized server and stochasticity into our federated ADMM framework. Specifically, we carefully consider the movement of the server, which is realized by using a dynamic Markov matrix. This is a significant novelty and challenge in the proof, as it is the first method introduced in federated learning that considers this type of server movement. By introducing assumptions on the dynamic Markov matrix and accounting for the dynamic behavior of the server, we can guarantee the convergence of our algorithm under certain conditions. While a few existing works address the convergence rates and properties of LADMM algorithms [142], they are not directly applicable to our random-walk mobilization and stochastic-update setting, and are not readily adaptable to the requirements of federated learning. As such, our work represents a significant contribution to the field and provides valuable insights into the convergence properties of stochastic ADMM-based federated learning algorithms. We believe that our novel approach and careful consideration of the movement of the server will inspire further research and development in this area, leading to more effective and efficient federated learning algorithms. The proof sketch of the Convergence Theorem (Theorem 3.9) is as follows.

Proof. The proof sketch is summarized as follows.

1. Under Assumption 3.3, the sequence created by the RWSADMM, i.e., $(\mathbf{y}^k, \mathbf{X}^k, \mathbf{Z}^k)_{k>T}$, satisfies

$$\left\|\mathbf{Z}^{k+1}-\mathbf{Z}^{k}\right\|=\sum\left\|\mathbf{z}_{i_k}^{k+1}-\mathbf{z}_{i_k}^{k}\right\|\leq L\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|$$

2. Recall L_β^k defined in Eq. (3.26). For $k\geq 0$, the RWSADMM iterates satisfy

$$L_\beta^k-L_\beta\left(\mathbf{y}^{k+1},\mathbf{X}^k;\mathbf{Z}^k\right)\geq\left\|\mathbf{y}^k-\mathbf{y}^{k+1}\right\|^2$$

3. Under Assumption 3.3, for $\beta>L$ and $\forall k>T$,

$$L_\beta(\mathbf{y}^{k+1},\mathbf{X}^k;\mathbf{Z}^k)-L_\beta^{k+1}\geq\frac{\beta-L}{2n}\left\|\mathbf{X}^k-\mathbf{X}^{k+1}\right\|^2-\frac{L^2}{n\beta}\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|^2$$

4. Recall M_β^k defined in Eq. (3.26); under Assumption 3.3, for $\beta>2L^2+L+2$ and $\forall k>T$,

$$M_\beta^k-M_\beta^{k+1}\geq\frac{\beta}{2}\left\|\mathbf{y}^k-\mathbf{y}^{k+1}\right\|^2+\frac{1}{n}\left\|\mathbf{X}^k-\mathbf{X}^{k+1}\right\|^2+\frac{L^2}{2n}\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|^2 \quad \text{(A.1)}$$

5. For $\beta>2L^2+L+2$, RWSADMM ensures a lower bounded sequence $(M_\beta^k)_{k\geq 0}$.

□

The proof details are provided in the following. Several steps are taken to prove Lemma 3.8. We introduce several Lemmas to represent these steps (Lemma A.1-A.5). Lemma A.1 shows that the update of the primal variable can bound the update on the dual variable.

Lemma A.1. *Under Assumption 3.3, the sequence created by RWSADMM,* $(\mathbf{y}^k,\mathbf{X}^k,\mathbf{Z}^k)_{k>T}$*, satisfies,*

$$\mathbb{E}\left\|\mathbf{Z}^{k+1}-\mathbf{Z}^{k}\right\|=\mathbb{E}\left\|\mathbf{z}_{i_k}^{k+1}-\mathbf{z}_{i_k}^{k}\right\|\leq L\cdot\mathbb{E}\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\| \quad \text{(A.2)}$$

Proof. Note that client i_k is activated at iteration k. Denote $\mathbf{x}_i^k$, $\mathbf{z}_i^k$, and $\mathbf{x}_i^k$ as the three groups of variables owned by any client i ($1\leq\forall i\leq n$) at iteration k. Under the Assumption 3.5, the optimality condition of $\mathbf{X}$ update for $i=i_k$ implies that

$$\mathbb{E}_\xi[sgn(\mathbf{t}')(g_i(\mathbf{x}_i^k,\xi)+\boldsymbol{\varepsilon}_i^k-\mathbf{z}_i^k)+\beta(\mathbf{y}^{k+1}-\mathbf{x}_{i_k}^{k+1})]=0 \quad \text{(A.3)}$$

Substituting Eq. (A.3) into Eq. (3.15) yields

$$\mathbb{E}_\xi[g_i(\mathbf{x}_i^k,\xi)]=\mathbb{E}_\xi[\mathbf{z}_{i_k}^{k+1}],\ \text{for } i=i_k \quad \text{(A.4)}$$

Hence, for $i = i_k$, we have

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{z}_i^{k+1} - \mathbf{z}_i^k\right\| &\overset{(a)}{=} \mathbb{E}\left\|\mathbf{z}_i^{k+1} - \mathbf{z}_i^{\tau(k,i)+1}\right\| \\
&\overset{(A.4)}{=} \left\|\mathbb{E}_\xi(g_i(\mathbf{x}_i^k, \xi) + \epsilon^{k+1}) - \mathbb{E}_\xi(g_i(\mathbf{x}_i^{\tau(k,i)}, \xi) + \epsilon^k)\right\| \\
&\overset{(b)}{=} \left\|\mathbb{E}_\xi(g_i(\mathbf{x}_i^{\tau(k,i)+1}, \xi)) - \mathbb{E}_\xi(g_i(\mathbf{x}_i^{\tau(k,i)}, \xi))\right\| \\
&\overset{(3.18\&3.21)}{\leq} L \cdot \mathbb{E}_\xi\left\|\mathbf{x}_i^{\tau(k,i)+1} - \mathbf{x}_i^{\tau(k,i)}\right\|
\end{aligned} \tag{A.5}$$

where $\tau(k,i)$ is defined in Eq. (3.25). The equality (a) holds because $\mathbf{z}_i^k = \mathbf{z}_i^{\tau(k,i)+1}$, and equality (b) holds because $\mathbf{x}_i^k = \mathbf{x}_i^{\tau(k,i)+1}$. On the other hand, when $i \neq i_k$, agent i is not activated at iteration k, so $\mathbb{E}_\xi[\|\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k\|] = L\mathbb{E}_\xi[\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\|] = 0$. Therefore, we have the proof of Eq. (A.2). □

Lemma A.2 shows that the $\mathbf{y}$-update in RWSADMM provides a sufficient descent of the augmented Lagrangian.

Lemma A.2. *Recall L_β^k defined in Eq.* (3.26). *For $k \geq 0$, RWSADMM iterates satisfies*

$$L_\beta^k - L_\beta\left(\mathbf{y}^{k+1}, \mathbf{X}^k; \mathbf{Z}^k\right) \geq \left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 \tag{A.6}$$

Proof. We rewrite the augmented Lagrangian function L in Eq. (3.8) by adding and subtracting the term $\|\mathbf{Z}\|^2/2\beta$ to the RHS of the equation and rearranging the terms. One can obtain

$$L_\beta\left(\mathbf{Y}, \mathbf{X}; \mathbf{Z}\right) = \frac{1}{n}\left(F(\mathbf{X}) + \frac{\beta}{2}\left\||\mathbf{1} \otimes \mathbf{y} - \mathbf{X}| - \boldsymbol{\epsilon} + \frac{\mathbf{Z}}{\beta}\right\|^2 - \frac{\|\mathbf{Z}\|^2}{2\beta}\right) \tag{A.7}$$

So, the augmented Lagrangian update is

$$\begin{aligned}
L_\beta^k - L_\beta\left(\mathbf{y}^{k+1}, \mathbf{X}^k; \mathbf{Z}^k\right) &= \frac{\beta}{2n}\left\||\mathbf{1} \otimes \mathbf{y}^k - \mathbf{X}^k| - \boldsymbol{\epsilon}^k + \frac{\mathbf{Z}^k}{\beta}\right\|^2 - \frac{\beta}{2n}\left\||\mathbf{1} \otimes \mathbf{y}^{k+1} - \mathbf{X}^k| - \boldsymbol{\epsilon}^k + \frac{\mathbf{Z}^k}{\beta}\right\|^2 \\
&\overset{(a)}{=} \frac{\beta}{2n}\sum_{i=1}^n \left(\left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 + 2\left\langle |\mathbf{y}^{k+1} - \mathbf{x}_i^k| - \boldsymbol{\epsilon}^k + \frac{\mathbf{z}_i^k}{\beta}, \mathbf{y}^k - \mathbf{y}^{k+1}\right\rangle\right) \geq \frac{\beta}{2}\left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 - \left\langle \mathbf{d}^k, \mathbf{y}^k - \mathbf{y}^{k+1}\right\rangle
\end{aligned} \tag{A.8}$$

The equality (a) is achieved by applying the cosine identity $\|b + c\|^2 - \|a + c\|^2 = \|b - a\|^2 + 2 < a + c, b - a >$, and $\mathbf{d}^k$ is defined as

$$\mathbf{d}^k := -\frac{\beta}{n}\sum_{i=1}^{n}\left(\left|\mathbf{y}^{k+1}-\mathbf{x}_i^k\right|-\boldsymbol{\epsilon}^k+\frac{\mathbf{z}_i^k}{\beta}\right)=0 \tag{A.9}$$

which results from $\mathbf{y}-$update in the RWSADMM algorithm. □

In Lemma A.3, the lower bound of descent in the augmented Lagrangian over the updates of $\mathbf{X}$ and $\mathbf{Z}$ is derived.

Lemma A.3. *Recall L_β^k defined in Eq. 3.26. Under Assumption 3.3, for $\beta > L$ and $\forall k > T$,*

$$\mathbb{E}[L_\beta(\mathbf{y}^{k+1},\mathbf{X}^k;\mathbf{Z}^k)-L_\beta^{k+1}] \quad \geq \frac{\gamma-L}{2n}\mathbb{E}\left\|\mathbf{X}^k-\mathbf{X}^{k+1}\right\|^2-\frac{L^2}{n\beta}\mathbb{E}\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|^2 \tag{A.10}$$

Proof. For the augmented Lagrangian L_β, we derive

$$
\begin{aligned}
&L_\beta(\mathbf{y}^{k+1}, \mathbf{X}^k; \mathbf{Z}^k) - L_\beta^{k+1} \\
&= \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) + \left\langle \mathbf{z}_{i_k}^k, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle + \frac{\beta}{2}\left\| \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\|^2 \\
&- f_{i_k}(\mathbf{x}_{i_k}^{k+1}) - \left\langle \mathbf{z}_{i_k}^{k+1}, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - \frac{\beta}{2}\left\| \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\|^2) \\
&= \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) - \left\langle \mathbf{z}_{i_k}^{k+1}, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) \\
&+ \left\langle \mathbf{z}_{i_k}^k, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - \frac{\beta}{2}\left[\left\| \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\|^2 - \left\| \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\|^2 \right]) \\
&\overset{(a)}{=} \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) - \left\langle \mathbf{z}_{i_k}^{k+1}, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) + \left\langle \mathbf{z}_{i_k}^k, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle \\
&- \frac{\beta}{2}\left[\left\| \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\|^2 + 2\left\langle \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}, \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\rangle \right]) \\
&= \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) - \left\langle \mathbf{z}_{i_k}^{k+1}, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) \\
&+ \left\langle \mathbf{z}_{i_k}^k, \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^k\right| - \boldsymbol{\varepsilon}_{i_k}^k \right\rangle - \frac{\beta}{2}\left\| \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\|^2 \\
&- \beta\left\langle \left|\mathbf{y}^{k+1} - \mathbf{x}_{i_k}^{k+1}\right| - \boldsymbol{\varepsilon}_{i_k}, \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\rangle) \\
&\overset{(b)}{=} \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) + \left\langle \mathbf{z}_{i_k}^k, \frac{\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k}{\beta} + \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\rangle - \frac{\beta}{2}\left\| \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\|^2 \\
&- \left\langle \mathbf{z}_{i_k}^{k+1}, \frac{\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k}{\beta} \right\rangle - \beta\left\langle \frac{\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k}{\beta}, \mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k \right\rangle) \\
&= \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) + \frac{\mathbf{z}_{i_k}^{k+1}\mathbf{z}_{i_k}^k}{\beta} - \frac{(\mathbf{z}_{i_k}^k)^2}{\beta} + \mathbf{z}_{i_k}^k(\mathbf{x}_{i_k}^{k+1} - \mathbf{x}_{i_k}^k) - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) \\
&+ \frac{\beta}{2}\left\| \mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1} \right\|^2 - \frac{(\mathbf{z}_{i_k}^{k+1})^2}{\beta} + \frac{\mathbf{z}_{i_k}^{k+1}\mathbf{z}_{i_k}^k}{\beta} - (\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k)(\mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1})) \\
&= \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) - \frac{1}{\beta}(\mathbf{z}_{i_k}^k)^2 + (\mathbf{z}_{i_k}^{k+1})^2 - 2(\mathbf{z}_{i_k}^{k+1}\mathbf{z}_{i_k}^k) \\
&+ \frac{\beta}{2}\left\| \mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1} \right\|^2 - \mathbf{z}_{i_k}^{k+1}(\mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1})) \\
&\overset{(c)}{=} \frac{1}{n}(f_{i_k}(\mathbf{x}_{i_k}^k) + \frac{\beta}{2}\left\| \mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1} \right\|^2 - f_{i_k}(\mathbf{x}_{i_k}^{k+1}) - \frac{1}{\beta}\left\| \mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k \right\|^2 - \left\langle \mathbf{z}_{i_k}^{k+1}, \mathbf{x}_{i_k}^k - \mathbf{x}_{i_k}^{k+1} \right\rangle)
\end{aligned}
\tag{A.11}
$$

Where equality (a) holds due to the cosine identity $\|b+c\|^2 - \|a+c\|^2 = \|b-a\|^2 + 2 < a+c, b-a >$, equality (b) holds because of $\mathbf{y}-$update in RWSADMM, and equality (c) holds due to recursion of $\mathbf{y}-$update in RWSADMM. Next, we apply the stochastic property to Eq. (A.11) using Eq. (A.4),

$$
\begin{aligned}
&\mathbb{E}[L_\beta(\mathbf{y}^{k+1},\mathbf{X}^k;\mathbf{Z}^k)-L_\beta^{k+1}]\\
&=\frac{1}{n}(\mathbb{E}_\xi[f_{i_k}(\mathbf{x}_{i_k}^k)-f_{i_k}(\mathbf{x}_{i_k}^{k+1})]-<E_\xi(g_i(\mathbf{x}_i^k,\xi)+\epsilon^k),\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}>\\
&+\frac{\beta}{2}E_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2-\frac{1}{\beta}\mathbb{E}_\xi\left\|\mathbf{z}_{i_k}^{k+1}-\mathbf{z}_{i_k}^k\right\|^2)\\
&\overset{(a)}{\geq}\frac{1}{n}\left(-\frac{L}{2}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2+\frac{\beta}{2}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2-\frac{1}{\beta}\mathbb{E}_\xi\left\|\mathbf{z}_{i_k}^{k+1}-\mathbf{z}_{i_k}^k\right\|^2\right)\\
&\overset{(b)}{\geq}\frac{1}{n}(-\frac{L}{2}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2+\frac{\beta}{2}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2-\frac{L^2}{\beta}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^{\tau(k,i_k)+1}-\mathbf{x}_{i_k}^{\tau(k,i_k)}\right\|^2)\\
&=\frac{\beta-L}{2n}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^k-\mathbf{x}_{i_k}^{k+1}\right\|^2-\frac{L^2}{n\beta}\mathbb{E}_\xi\left\|\mathbf{x}_{i_k}^{\tau(k,i_k)+1}-\mathbf{x}_{i_k}^{\tau(k,i_k)}\right\|^2
\end{aligned}
\tag{A.12}
$$

Where inequality (a) is achieved using Assumption 3.3 (Eq. (3.19)), and inequality (b) is due to Lemma A.1 (Eq. (A.2)). □

In Lemma A.4, the sufficient descent in Lyapunov functions is established.

Lemma A.4. *Recall M_β^k defined in Eq.* (3.26)*; under Assumption 3.3, for $\beta > 2L^2+L+2$ and $\forall k > T$,*

$$
M_\beta^k-M_\beta^{k+1}\geq\frac{\beta}{2}\left\|\mathbf{y}^k-\mathbf{y}^{k+1}\right\|^2+\frac{1}{n}\left\|\mathbf{X}^k-\mathbf{X}^{k+1}\right\|^2+\frac{L^2}{2n}\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|^2
\tag{A.13}
$$

Proof. Using Eq. (3.26), we can attain

$$
\begin{aligned}
&M_\beta^k-M_\beta^{k+1}\\
&=L_\beta^k-L_\beta^{k+1}+\frac{L^2}{n}(\left\|\mathbf{x}_i^{\tau(k,i_k)+1}-\mathbf{x}_i^{\tau(k,i_k)}\right\|^2-\left\|\mathbf{x}_i^{\tau(k+1,i_k)+1}-\mathbf{x}_i^{\tau(k+1,i_k)}\right\|^2)\\
&\overset{(a)}{=}L_\beta^k-L_\beta^{k+1}+\frac{L^2}{n}(\left\|\mathbf{x}_{i_k}^{\tau(k,i_k)+1}-\mathbf{x}_{i_k}^{\tau(k,i_k)}\right\|^2-\left\|\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^k\right\|^2)\\
&=L_\beta^k-L_\beta^{k+1}+\frac{L^2}{n}(\left\|\mathbf{X}^{\tau(k,i_k)+1}-\mathbf{X}^{\tau(k,i_k)}\right\|^2-\left\|\mathbf{X}^{k+1}-\mathbf{X}^k\right\|^2)
\end{aligned}
\tag{A.14}
$$

where inequality (a) is due to the following property,

$$
x_i^{\tau(k+1,i)+1}-x_i^{\tau(k+1,i)}=\begin{cases}x_i^{k+1}-x_i^k, & i=i_k\\ x_i^{\tau(k,i)+1}-x_i^{\tau(k,i)}, & otherwise\end{cases}
\tag{A.15}
$$

Substituting Eq. (A.1), one can obtain

$$\begin{aligned} M_\beta^k - M_\beta^{k+1} \geq L_\beta\left(\mathbf{y}^{k+1}, \mathbf{X}^k; \mathbf{Z}^k\right) - L_\beta^{k+1} + \left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 \\ + \frac{L^2}{n}\left(\left\|\mathbf{X}^{\tau(k,i_k)+1} - \mathbf{X}^{\tau(k,i_k)}\right\|^2 - \left\|\mathbf{X}^{k+1} - \mathbf{X}^k\right\|^2\right) \end{aligned} \tag{A.16}$$

One can also substitute Eq. (A.1) into Eq. (A.16), which leads to

$$\begin{aligned} & M_\beta^k - M_\beta^{k+1} \\ & \geq \frac{\beta - L}{2n}\left\|\mathbf{X}^k - \mathbf{X}^{k+1}\right\|^2 - \frac{L^2}{n\beta}\left\|\mathbf{X}^{\tau(k,i_k)+1} - \mathbf{X}^{\tau(k,i_k)}\right\|^2 \\ & + \left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 + \frac{L^2}{n}\left(\left\|\mathbf{X}^{\tau(k,i_k)+1} - \mathbf{X}^{\tau(k,i_k)}\right\|^2 - \left\|\mathbf{X}^{k+1} - \mathbf{X}^k\right\|^2\right) \\ & = \frac{\beta - L - 2L^2}{2n}\left\|\mathbf{X}^k - \mathbf{X}^{k+1}\right\|^2 + \frac{L^2(\beta-1)}{n\beta}\left\|\mathbf{X}^{\tau(k,i_k)+1} - \mathbf{X}^{\tau(k,i_k)}\right\|^2 + \left\|\mathbf{y}^k - \mathbf{y}^{k+1}\right\|^2 \end{aligned} \tag{A.17}$$

Using $\frac{\beta}{2} - \frac{L}{2} - L^2 \geq 1$, $1 - \frac{1}{\beta} > \frac{1}{2}$, and $\beta < 2$, we complete the proof of Eq. (A.13). □

Lemma A.5 states that the Lyapunov function M_β^k is lower bounded.

Lemma A.5. *For $\beta > 2L^2 + L + 2$, RWSADMM ensures a lower bounded sequence $(M_\beta^k)_{k\geq 0}$ in expectation.*

Proof. For $k > T$, we have

$$
\begin{aligned}
&\mathbb{E}[M_\beta^k] \\
&= \mathbb{E}[L_\beta^k] + \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left\|\mathbf{x}_i^{\tau(k,i_k)+1} - \mathbf{x}_i^{\tau(k,i_k)}\right\|^2 \\
&= \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left(f_j(\mathbf{x}_j^k) + \left\langle \mathbf{z}_j^k, \left|\mathbf{y}^k - \mathbf{x}_j^k\right| - \boldsymbol{\varepsilon}_j^k \right\rangle \right) + \frac{\beta}{2n}\sum_{j=1}^n \mathbb{E}\left\|\mathbf{y}_j^k - \mathbf{X}_j^k\right\|^2 \\
&+ \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left\|\mathbf{x}_i^{\tau(k,i_k)+1} - \mathbf{x}_i^{\tau(k,i_k)}\right\|^2 \\
&\overset{(a)}{=} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left(f_j(\mathbf{x}_j^k) + \left\langle \mathbb{E}_\xi(g_j(\mathbf{x}_j^{\tau(k,j)}, \xi)), \mathbf{y}^k - \mathbf{x}_j^k \right\rangle \right) + \frac{\beta}{2n}\sum_{j=1}^n \mathbb{E}\left\|\mathbf{y}_j^k - \mathbf{X}_j^k\right\|^2 \\
&\qquad + \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left\|\mathbf{x}_i^k - \mathbf{x}_i^{\tau(k,i_k)}\right\|^2 \\
&\overset{(b)}{\geq} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left(f_j(\mathbf{y}^\mathbf{k}) + \left\langle \mathbb{E}_\xi(g_j(\mathbf{x}_j^{\tau(k,j)}, \xi)) - \mathbb{E}_\xi(g_j(\mathbf{x}_i^k, \xi)), \mathbf{y}^k - \mathbf{x}_j^k \right\rangle \right) \\
&+ \frac{\beta - L}{2n}\sum_{j=1}^n \mathbb{E}\left\|\mathbf{y}_j^k - \mathbf{X}_j^k\right\|^2 + \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left\|\mathbf{x}_i^k - \mathbf{x}_i^{\tau(k,i_k)}\right\|^2 \\
&\overset{(c)}{\geq} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left(f_j(\mathbf{y}^\mathbf{k}) + \left\|\mathbb{E}_\xi(g_j(\mathbf{x}_i^{\tau(k,j)}, \xi)) - \mathbb{E}_\xi(g_j(\mathbf{x}_i^k, \xi)\right\|^2 \right) \\
&+ \frac{\beta - L - 2}{2n}\mathbb{E}\left\|\mathbf{1}\otimes\mathbf{y}^k - \mathbf{X}^k\right\|^2 + \frac{L^2}{n}\sum_{i=1}^n \mathbb{E}\left\|\mathbf{x}_i^k - \mathbf{x}_i^{\tau(k,i_k)}\right\|^2 \\
&\overset{(d)}{\geq} \min_{\mathbf{y}} \left\{ \frac{1}{n}\sum_{j=1}^n \mathbb{E} f_j(\mathbf{y}^k) \right\} + \frac{2L^2}{2n}\mathbb{E}\left\|\mathbf{1}\otimes\mathbf{y}^k - \mathbf{X}^k\right\|^2 \\
&\overset{(e)}{\geq} -\infty
\end{aligned}
\tag{A.18}
$$

where (a) holds due to Eq. (A.4). (b) holds because f_j is Lipschitz differentiable (Eq. (3.23)). (c) holds due to Young's inequality. (d) follows from the assumption $\beta > 2L^2+L+2$ and the Lipschitz smoothness of each f_j, and (e) holds due to Assumption 3.2. Therefore, M_β^k is bounded from below in expectation. □

Now we can prove Lemma 3.8 with the above lemmas.

Proof. Recall that the maximum hitting time T is almost surely finite. For Statement 1, the monotonicity of Lyapunov function $(M_\beta^k)_{k>T}$ in Lemma A.4 and their lower boundedness in Lemma A.5 ensure convergence of $(M_\beta^k)_{k\geq 0}$. For Statement 2, consider Statement 1 and the lower boundedness of $(M_\beta^k)_{k>T}$ in Lemma A.5 (Eq. (A.18)). $\frac{1}{n}F(\mathbf{y}^k)$

is upper bounded by $\max\{\max_{t\in\{0,\ldots,T\}}\{\frac{1}{n}F(\mathbf{y}^t)\}, M_\beta^{T+1}\}$; and $\|\mathbf{1}\otimes\mathbf{y}_j^k-\mathbf{X}_j^k\|^2$ is upper bounded by $\max\{\max_{\{t\in 0,\ldots,T\}}\{\|\mathbf{1}\otimes\mathbf{y}_j^t-\mathbf{X}_j^t\|^2\}, L_\beta^{T+1}\}$. By Assumption 3.2, the sequence $\{\mathbf{y}|k=\{0,1,\ldots,\}\}$ is bounded. The boundedness of $\|\mathbf{1}\otimes\mathbf{y}_j^k-\mathbf{X}_j^k\|^2$ further leads to that of $\{\mathbf{X}_j^k|k=\{0,1,\ldots,\}\}$. Finally, Eq. (A.4) and Assumption 3.3 ensure $(\mathbf{Z}^k)$ is bounded as well. Altogether, $(\mathbf{y}^k,\mathbf{X}^k,\mathbf{Z}^k)$ is bounded. □

Based on Lemma 3.8, the convergence of the subgradients of L_β^k can be established as follows.

Lemma A.6. *With Assumption 3.6 and β given in Lemma 3.8, for any given subsequence (including the whole sequence) with its index $(k_s)_{s\geq 0}$, there exists a sequence $(g^k)_{k\geq 0}$ with $(g^k)\in\partial L_\beta^{k+1}$ containing an almost surely convergent subsequence $(g^{k_{s_j}})_{j\geq 0}$, that is,*

$$Pr\left(\lim_{j\to\infty}\left\|g^{k_{s_j}}=0\right\|\right)=1$$

Proof. The proof sketch is summarized as follows.

1. We construct the sequence $g^k\in\partial L_\beta^{k+1}$ and show that its subvector $q_i^k:=\left(g_{\mathbf{y}^i}^k,g_{\mathbf{x}^i}^k,g_{\mathbf{z}^i}^k\right)$ satisfies $\lim_{k\to\infty}\mathbb{E}\left\|q_{i_k}^{k-\tau(\delta)-1}\right\|^2=0$, where the mixing time $\tau(\delta)$ is defined in Eq. (3.6).

2. For $k\geq 0$, define the filtration of sigma algebras:
   $\chi^k=\sigma\left(\mathbf{y}^0,\ldots,\mathbf{y}^k,\mathbf{X}^0,\ldots,\mathbf{X}^k,\mathbf{Z}^0,\ldots,\mathbf{Z}^k,i_0,\ldots,i_k\right)$. We show that
   $\mathbb{E}\left(\left\|q_{i_k}^{k-\tau(\delta)-1}\right\|^2\middle|\chi^{k-\tau(\delta)}\right)\geq(1-\delta)\pi_*\left\|g^{k-\tau(\delta)-1}\right\|^2$,
   where π_* is the minimal value in the Markov chain's stationary distribution. From this bound and the result in Step 1, we can get $\lim_{k\to\infty}\left\|g^k\right\|=0$.

3. From the result in Step 2, we use the Borel-Cantelli lemma [143] to obtain an almost unquestionably convergent subsubsequence of g^k.

□

The details of these steps are given as follows.

Proof. First, recall Lemma A.4 and $T<\infty$, we have

$$\sum_{k=0}^{\infty}\left(\mathbb{E}\left\|\mathbf{y}^k-\mathbf{y}^{k+1}\right\|^2+\mathbb{E}\left\|\mathbf{X}^k-\mathbf{X}^{k+1}\right\|^2+\mathbb{E}\left\|\mathbf{X}^{\tau(k,i_k)}-\mathbf{X}^{\tau(k,i_k)+1}\right\|_2^2\right)<+\infty \quad \text{(A.19)}$$

Hence, by Lemma A.4, one can infer

$$\sum_{k=0}^{\infty}\left(\mathbb{E}\left\|\mathbf{y}^{k}-\mathbf{y}^{k+1}\right\|^{2}+\mathbb{E}\left\|\mathbf{X}^{k}-\mathbf{X}^{k+1}\right\|^{2}+\mathbb{E}\left\|\mathbf{Z}^{k}-\mathbf{Z}^{k+1}\right\|_{2}^{2}\right)<+\infty \tag{A.20}$$

Step 1: The proof starts with computing the subgradients of the augmented Lagrangian Eq. (A.7) with the updates in RWSADMM,

$$\frac{\partial L_{\beta}^{k+1}}{\partial \mathbf{y}} \ni -\frac{\beta}{n}(\mathbf{x}_{i_k}^{k+1}-\mathbf{x}_{i_k}^{k})+\frac{1}{n}(\mathbf{z}_{i_k}^{k+1}-\mathbf{z}_{i_k}^{k}) =: w^{k} \tag{A.21}$$

$$\boldsymbol{\nabla}_{\mathbf{x}_j} L_{\beta}^{k+1}=\frac{1}{n}\left(\boldsymbol{\nabla} f_j(\mathbf{x}_j^{k+1})-\mathbf{z}_j^{k+1}+\beta(\mathbf{x}_j^{k+1}-\mathbf{y}^{k+1})\right) \tag{A.22}$$

$$\boldsymbol{\nabla}_{\mathbf{z}_j} L_{\beta}^{k+1}=\frac{1}{n}(\mathbf{y}^{k+1}-\mathbf{x}_j^{k+1}) \tag{A.23}$$

We define g^k and q_i^k as

$$g^{k}:=\begin{bmatrix} w^{k} \\ \boldsymbol{\nabla}_{\mathbf{X}} L_{\beta}^{k+1} \\ \boldsymbol{\nabla}_{\mathbf{Z}} L_{\beta}^{k+1}\end{bmatrix}, q_i^{k}:=\begin{bmatrix} w^{k} \\ \boldsymbol{\nabla}_{\mathbf{x}_i} L_{\beta}^{k+1} \\ \boldsymbol{\nabla}_{\mathbf{z}_i} L_{\beta}^{k+1}\end{bmatrix} \tag{A.24}$$

where $i \in \mathbf{V}$ is the index of the agent and g^k is the gradient of L_k^{β}. For $\delta \in (0,1)$ and $k \geq \tau(\delta)+1$,

$$\left\|q_{i_k}^{k-\tau(\delta)-1}\right\|^{2}=\left\|q_{i_k}^{k-\tau(\delta)-1}-q_{i_k}^{k}+q_{i_k}^{k}\right\|^{2} \overset{(a)}{\leq} 2\underbrace{\left\|q_{i_k}^{k-\tau(\delta)-1}-q_{i_k}^{k}\right\|^{2}}_{\text{A}}+2\underbrace{\left\|q_{i_k}^{k}\right\|^{2}}_{\text{B}} \tag{A.25}$$

We upper bound A and B separately. A has three parts corresponding to the three parts of g. Its first part is

$$\begin{aligned}
&\left\|w^{k-\tau(\delta)-1}-w^{k}\right\|^{2} \\
&\leq 2\left\|w^{k-\tau(\delta)-1}\right\|^{2}+2\left\|w^{k}\right\|^{2} \\
&\overset{\text{(A.21)}}{\leq} \frac{4}{n^{2}}\Big(\beta^{2}\left\|\mathbf{X}^{k+1}-\mathbf{X}^{k}\right\|^{2}+\beta^{2}\left\|\mathbf{X}^{k-\tau(\delta)}-\mathbf{X}^{k-\tau(\delta)-1}\right\|^{2} \\
&+\left\|\mathbf{Z}^{k+1}-\mathbf{Z}^{k}\right\|^{2}+\left\|\mathbf{Z}^{k-\tau(\delta)}-\mathbf{Z}^{k-\tau(\delta)-1}\right\|^{2}\Big)
\end{aligned} \tag{A.26}$$

Then by Eq. (A.22), we bound the 2nd part of A

$$
\begin{aligned}
&\left\|\boldsymbol{\nabla}_{\mathbf{x}_{i_k}} L_\beta^{k-\tau(\delta)-1} - \boldsymbol{\nabla}_{\mathbf{x}_{i_k}} L_\beta^{k+1}\right\|^2 \\
&\overset{(a)}{\leq} \frac{4L^2+4\beta^2}{n^2}\left\|\mathbf{x}_{i_k}^{k-\tau(\delta)-1} - \mathbf{x}_{i_k}^{k+1}\right\|^2 + \frac{4}{n^2}\left\|\mathbf{z}_{i_k}^{k-\tau(\delta)-1} - \mathbf{z}_{i_k}^{k+1}\right\|^2 \\
&+ \frac{4\beta^2}{n^2}\left\|\mathbf{y}^{k-\tau(\delta)-1} - \mathbf{y}^{k+1}\right\|^2 \\
&\leq \frac{D}{n^2}\sum_{t=k-\tau(\delta)-1}^{k}\left(\left\|\mathbf{y}^t - \mathbf{y}^{t+1}\right\|^2 + \left\|\mathbf{X}^t - \mathbf{X}^{t+1}\right\|^2 + \left\|\mathbf{Z}^t - \mathbf{Z}^{t+1}\right\|^2\right)
\end{aligned}
\tag{A.27}
$$

where $D = (\tau(\delta)+2)(4+4\beta^2+4L^2)$, and (a) uses the inequality of arithmetic and geometric means and Lipschitz differentiability of f_j in Assumption 3.3. From Eq. (A.23). The third part of A can be bounded as

$$
\begin{aligned}
&\left\|\boldsymbol{\nabla}_{\mathbf{z}_{i_k}} L_\beta^{k-\tau(\delta)-1} - \boldsymbol{\nabla}_{\mathbf{z}_{i_k}} L_\beta^{k+1}\right\|^2 \\
&\leq \frac{2}{n^2}\left(\left\|\mathbf{y}^{k-\tau(\delta)-1} - \mathbf{y}^{k+1}\right\|^2 + \left\|\mathbf{x}_{i_k}^{k-\tau(\delta)-1} - \mathbf{x}_{i_k}^{k+1}\right\|^2\right) \\
&\leq \frac{2(\tau(\delta)+2)}{n^2}\sum_{t=k-\tau(\delta)-1}^{k}\left(\left\|\mathbf{y}^t - \mathbf{y}^{t+1}\right\|^2 + \left\|\mathbf{X}^t - \mathbf{X}^{t+1}\right\|^2\right)
\end{aligned}
\tag{A.28}
$$

Plugging Eq. (A.26), Eq. (A.27), and Eq. (A.28) into term A, we get a constant $C_1 \sim O(\frac{\tau(\delta)+1}{n^2})$, depending on $\tau(\delta)$, β, L, and n, such that

$$
A \leq C_1 \sum_{t=k-\tau(\delta)-1}^{k}\left(\left\|\mathbf{y}^t - \mathbf{y}^{t+1}\right\|^2 + \left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|^2 + \left\|\mathbf{z}^t - \mathbf{z}^{t+1}\right\|^2\right) \tag{A.29}
$$

To bound the term B, using Eq. (A.23) and $\mathbf{Z}$-update (Eq. (3.15)), we have

$$
\boldsymbol{\nabla}_{\mathbf{z}_{i_k}} L_\beta^{k+1} = \frac{1}{n\beta}(\mathbf{z}_{i_k}^{k+1} - \mathbf{z}_{i_k}^k) \tag{A.30}
$$

Applying Eq. (A.22) and Eq. (A.4), we drive $\boldsymbol{\nabla}_{\mathbf{x}_{i_k}} L_\beta^{k+1}$:

$$
\boldsymbol{\nabla}_{\mathbf{x}_{i_k}} L_\beta^{k+1} = \frac{1}{n\beta}\left(\boldsymbol{\nabla} f_{i_k}(\mathbf{x}_{i_k}^{k+1}) - \boldsymbol{\nabla} f_{i_k}(\mathbf{x}_{i_k}^k) + \mathbf{z}_{i_k}^k - \mathbf{z}_{i_k}^{k+1}\right) \tag{A.31}
$$

So we have

$$
B \leq C_2\left(\left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|^2 + \left\|\mathbf{z}^t - \mathbf{z}^{t+1}\right\|^2\right) \tag{A.32}
$$

for a constant C_2 depending on L, β, and n, in the order of $O(\frac{1}{n^2})$. Then substituting Eq.

(A.29) and Eq. (A.32) into Eq. (A.25) and taking expectations, it yields

$$\begin{aligned}&\mathbb{E}\Big\|q_{i_k}^{k-\tau(\delta)-1}\Big\|^2\\&\leq C\sum_{t=k-\tau(\delta)-1}^{k}\left(\mathbb{E}\big\|\mathbf{y}^t-\mathbf{y}^{t+1}\big\|^2+\mathbb{E}\big\|\mathbf{x}^t-\mathbf{x}^{t+1}\big\|^2+\mathbb{E}\big\|\mathbf{z}^t-\mathbf{z}^{t+1}\big\|^2\right)\end{aligned}\tag{A.33}$$

where $C=C_1+C_2$, and $C\sim O(\frac{\tau(\delta)+1}{n^2})$. Recalling Eq. (A.20), we get the convergence

$$\lim_{k\to\infty}\mathbb{E}\Big\|q_{i_k}^{k-\tau(\delta)-1}\Big\|^2=0\tag{A.34}$$

which completes the proof of **Step 1**.

Step 2: We compute the expectation:

$$\begin{aligned}&\mathbb{E}\Big(\Big\|q_{i_k}^{k-\tau(\delta)-1}\Big\|^2\Big|\chi^{k-\tau(\delta)}\Big)\\&=\sum_{j=1}^{n}[\mathbf{P}(k)^{\tau(\delta)}]_{i_{k-\tau(\delta)},j}\Big(\Big\|\boldsymbol{\nabla}_{\mathbf{y}}L_\beta^{k-\tau(\delta)}\Big\|^2+\Big\|\boldsymbol{\nabla}_{\mathbf{x_j}}L_\beta^{k-\tau(\delta)}\Big\|^2\\&+\Big\|\boldsymbol{\nabla}_{\mathbf{z_j}}L_\beta^{k-\tau(\delta)}\Big\|^2\Big)\overset{(a)}{\geq}(1-\delta)\pi_*\big\|g^{k-\tau(\delta)-1}\big\|^2\end{aligned}\tag{A.35}$$

where (a) follows from Eq. (3.4) and the definition of g_k in Eq. (A.24). Then, with Eq. (A.33), one can derive

$$\lim_{k\to\infty}\mathbb{E}\big\|g^k\big\|^2=\lim_{k\to\infty}\mathbb{E}\big\|g^{k-\tau(\delta)-1}\big\|^2=0,\tag{A.36}$$

By the Schwarz inequality $(\mathbb{E}\big\|g^k\big\|)^2\leq\mathbb{E}\big\|g^k\big\|^2$, we have

$$\lim_{k\to\infty}\mathbb{E}\big\|g^k\big\|=0,\tag{A.37}$$

Step 3: By Markov's inequality, for each $\omega>0$, it holds that

$$Pr(\|g_k\|>\omega)\leq\frac{\mathbb{E}\big\|g^k\big\|}{\omega}\overset{Eq.(A.36)}{\Longrightarrow}\lim_{k\to\infty}Pr(\|g_k\|>\omega)=0\tag{A.38}$$

when a subsequence $(k_s)_{s\geq0}$ is provided, Eq. (A.37) implies,

$$Pr(\|g_{k_s}\|>\omega)=0\tag{A.39}$$

Then, for $j\in\mathbb{N}$, select $\omega=2^{-j}$ and we can find a nondecreasing subsubsequence (k_{s_j}),

such that

$$Pr(\|g_{k_s}\| > 2^{-j}) \leq 2^{-j}, \ \ \forall k_s \geq k_{s_j} \tag{A.40}$$

We have

$$\sum_{j=1}^{\infty} Pr(\|g_{k_s}\| > 2^{-j}) \leq \sum_{j=1}^{\infty} 2^{-j} = 1 \tag{A.41}$$

The Borel-Cantelli lemma yields

$$Pr\left(\limsup_{j}\{\|g_{k_s}\| > 2^{-j}\}\right) = 0 \tag{A.42}$$

and thus

$$Pr\left(\lim_{j}\left\|g_{k_{s_j}}\right\| = 0\right) = 1 \tag{A.43}$$

This completes **Step 3** and thus the entire **Lemma A.6**.

□

Proof of Convergence Finally, we present the proof of the convergence theorem (Theorem 3.9) as follows.

Proof. By statement 2 of Lemma 3.8, the sequence $(\mathbf{y}^k, \mathbf{X}^k, \mathbf{Z}^k)$ is bounded, so there exists a convergent subsequence $(\mathbf{y}^{k_s}, \mathbf{X}^{k_s}, \mathbf{Z}^{k_s})$ converging to a limit point $(\mathbf{y}^*, \mathbf{X}^*, \mathbf{Z}^*)$ as $s \to \infty$. By continuity, we have

$$L_\beta(\mathbf{y}^*, \mathbf{X}^*, \mathbf{Z}^*) = \lim_{s\to\infty} L_\beta(\mathbf{y}^{k_s}, \mathbf{X}^{k_s}, \mathbf{Z}^{k_s}) \tag{A.44}$$

Lemma A.6 finds a subsubsequence $g^{k_{s_j}} \in \partial L_\beta^{k+1}$ such that $Pr\big(\lim_{j\to\infty} \left\|g^{k_{s_j}}\right\| = 0\big) = 1$. By the definition of general subgradient ([144], def 8.3), we have $0 \in \partial L_\beta(\mathbf{y}^*, \mathbf{X}^*, \mathbf{Z}^*)$. Hence, Theorem 3.9 is proved. □

APPENDIX B

RWSADMM Convergence Rate Analysis

B.1 RWSADMM: Convergence Rate Analysis

The detailed proof of the convergence rate theorem is as follows.

Proof. It can be verified that under specific initialization, Eq. (A.4) holds for all $k \geq 0$. Consequently, Lemmas A.1-A.4 hold for all $k \geq 0$. For g^k defined in Eq. (A.24), Eq. (A.33) and Eq. (A.35) hold. Jointly applying Eq. (A.33) and Eq. (A.35), for any $k > \tau(\delta) + 1$, one has

$$\begin{aligned} &\mathbb{E}\big\|g^{k-\tau(\delta)-1}\big\|^2 \leq \frac{C}{(1-\delta)\pi_*} \\ &\sum_{t=k-\tau(\delta)-1}^{k} \big(\mathbb{E}\big\|\mathbf{y}^t - \mathbf{y}^{t+1}\big\|^2 + \mathbb{E}\big\|\mathbf{X}^t - \mathbf{X}^{t+1}\big\|^2 + \mathbb{E}\big\|\mathbf{Z}^t - \mathbf{Z}^{t+1}\big\|^2\big) \end{aligned} \tag{B.1}$$

According to Lemmas A.1 and A.4, for $k \geq \tau(\delta) + 1$, it holds,

$$\begin{aligned} &\sum_{t=k-\tau(\delta)-1}^{k} \big(\mathbb{E}\big\|\mathbf{y}^t - \mathbf{y}^{t+1}\big\|^2 + \mathbb{E}\big\|\mathbf{X}^t - \mathbf{X}^{t+1}\big\|^2 + \mathbb{E}\big\|\mathbf{Z}^t - \mathbf{Z}^{t+1}\big\|^2\big) \\ &\leq \max\big\{\frac{2}{\beta-\gamma}, (1+L^2)n\big\}\big(\mathbb{E}L_\beta^{k-\tau(\delta)-1} - \mathbb{E}L_\beta^{k+1}\big) \end{aligned} \tag{B.2}$$

It implies that for any $k \geq 0$, it holds

$$\mathbb{E}\big\|g^k\big\|^2 \leq C'\big(\mathbb{E}L_\beta^k - \mathbb{E}L_\beta^{k+\tau(\delta)+2}\big) \tag{B.3}$$

where $C' := \max\{\frac{2}{\beta-\gamma}, (1+L^2)n\}\frac{C}{(1-\delta)\pi_*}$. It can be verified that $C' = O(\frac{\tau(\delta)+1}{(1-\delta)n\pi_*})$. Let $\tau' := \tau(\delta) + 2$; for any $K > \tau'$, summing Eq. (B.3) over $k \in \{K - \tau', \ldots, \mathrm{mod}_{\tau'} K\}$ gives

$$\sum_{l=1}^{\lfloor \frac{K}{\tau'} \rfloor} \mathbb{E}\left\| g^{K-l\tau'} \right\|^2 \leq C' \left(\mathbb{E} L_\beta^{\mathrm{mod}_{\tau' K}} - \mathbb{E} L_\beta^K \right) \leq C' (L_\beta^0 - \underline{f}) \tag{B.4}$$

where the last inequality follows from the non-decreasing property of the sequence $(L_\beta^k)_{k\geq 0}$ and the fact that $(L_\beta^k)_{k\geq 0}$ is lower bounded by $\underline{f}$. According to Eq. (B.4),

$$\begin{aligned} \min_{k\leq K} \mathbb{E}\left\| g^k \right\|^2 &\leq \min_{1\leq l\leq \lfloor \frac{K}{\tau'} \rfloor} \mathbb{E}\left\| g^{K-l\tau'} \right\|^2 \\ &\leq \frac{1}{\lfloor \frac{K}{\tau'} \rfloor} \sum_{l=1}^{\lfloor \frac{K}{\tau'} \rfloor} \mathbb{E}\left\| g^{K-l\tau'} \right\|^2 \leq \frac{\tau' C'}{K-\tau'} (L_\beta^0 - \underline{f}) \\ &\leq \frac{C'(\tau'+1)}{K} (L_\beta^0 - \underline{f}) \end{aligned} \tag{B.5}$$

where the constant $C'(\tau'+1) = O\big(\frac{\tau(\delta)^2+1}{(1-\delta)n\pi_*}\big)$. □

We consider a reversible Markov chain on an undirected graph. Using the definition of $\tau(\delta)$ in Eq. (3.6) and setting $\delta = 1/2$, one has $\tau(\delta) \sim \frac{\ln n}{1-\lambda_2(P(k))}$. To guarantee $\min_{k\leq K} \mathbb{E}\left\| g^k \right\|^2 \leq \omega$, certain number of iterations is sufficient,

$$O\left(\frac{1}{\omega} . \left(\frac{\ln^2 n}{(1-\lambda_2(P(k)))^2} \right) + 1 \right) \tag{B.6}$$

BIBLIOGRAPHY

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.

[2] T. Li, A. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] W. Lim, N. Luong, D. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[4] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information Processing & Management*, vol. 59, no. 6, p. 103061, 2022.

[5] D. Handelman, C. Rivera, R. Amant, E. Holmes, A. Badger, and B. Yeh, "Adaptive human-robot teaming through integrated symbolic and subsymbolic artificial intelligence: preliminary results," in *SPIE*, vol. 12113, 2022, pp. 145–157.

[6] J. Gilmore, "Warfighter information Network–Tactical (WIN-T) increment 2," Director, Operational Test & Evaluation (DOT&E), U.S. Department of Defense, Tech. Rep., 2015.

[7] F. Sadikin and S. Kumar, "Zigbee IoT intrusion detection system: A hybrid approach with rule-based and machine learning anomaly detection." in *IoTBDS*, 2020, pp. 57–68.

[8] N. Mowla, N. Tran, I. Doh, and K. Chae, "Federated learning-based cognitive detection of jamming attack in flying ad-hoc network," *IEEE Access*, vol. 8, pp. 4338–4350, 2019.

[9] B. Li, Y. Jiang, W. Sun, W. Niu, and P. Wang, "FedVANET: Efficient federated learning with Non-IID data for vehicular ad hoc networks," in *IEEE GLOBECOM*, 2021, pp. 1–6.

[10] H. Ochiai, Y. Sun, Q. Jin, N. Wongwiwatchai, and H. Esaki, "Wireless ad hoc federated learning: A fully distributed cooperative machine learning," *arXiv preprint arXiv:2205.11779*, 2022.

[11] X. Liu, T. Yang, and B. Yan, "Internet of things for wildlife monitoring," in *IEEE/CIC ICCC Workshops*, 2015, pp. 62–66.

[12] V. Toldov, L. Clavier, and N. Mitton, "Multi-channel distributed MAC protocol for WSN-based wildlife monitoring," in *IEEE WiMob*, 2018, pp. 1–8.

[13] B. Arshad, J. Barthelemy, E. Pilton, and P. Perez, "Where is my deer?-wildlife tracking and counting via edge computing and deep learning," in *IEEE SENSORS*, 2020, pp. 1–4.

[14] C.-C. Kao, Y.-S. Lin, G.-D. Wu, and C.-J. Huang, "A comprehensive study on the internet of underwater things: applications, challenges, and channel models," *Sensors*, vol. 17, no. 7, p. 1477, 2017.

[15] N. Victor, M. Alazab, S. Bhattacharya, S. Magnusson, P. Maddikunta, K. Ramana, T. Gadekallu *et al.*, "Federated learning for IoUT: Concepts, applications, challenges and opportunities," *arXiv preprint arXiv:2207.13976*, 2022.

[16] L. Ahmed, K. Ahmad, N. Said, B. Qolomany, J. Qadir, and A. Al-Fuqaha, "Active learning based federated learning for waste and natural disaster image classification," *IEEE Access*, vol. 8, pp. 208 518–208 531, 2020.

[17] A. Imteaj, I. Khan, J. Khazaei, and M. Amini, "Fedresilience: A federated learning application to improve resilience of resource-constrained critical infrastructures," *Electronics*, vol. 10, no. 16, p. 1917, 2021.

[18] A. Bairwa and S. Joshi, "MLA-RPM: A machine learning approach to enhance trust for secure routing protocol in mobile ad hoc networks," *Int J Adv Sci Technol*, vol. 29, no. 04, pp. 11 265–11 274, 2020.

[19] D. Helbing and E. Pournaras, "Society: Build digital democracy," *Nature*, vol. 527, no. 7576, pp. 33–34, 2015.

[20] J. Gastil and R. Richards, "Embracing digital democracy: A call for building an online civic commons," *PS: Political Science & Politics*, vol. 50, no. 3, pp. 758–763, 2017.

[21] J. Mao, S. Shi, X. Wang, and H. Li, "3D object detection for autonomous driving: A comprehensive survey," *International Journal of Computer Vision*, vol. 131, no. 8, pp. 1909–1963, 2023.

[22] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *IEEE ICRA*, 2023, pp. 2774–2781.

[23] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *IEEE/CVF CVPR*, 2022, pp. 1080–1089.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE CVPR*, 2012, pp. 3354–3361.

[25] A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *IEEE/CVF CVPR*, 2019, pp. 12 697–12 705.

[26] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[27] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D object detection and tracking," in *IEEE/CVF CVPR*, 2021.

[28] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *IEEE/CVF CVPR*, 2020, pp. 11 621–11 631.

[29] Y. Zhang, J. Hou, and Y. Yuan, "A comprehensive study of the robustness for lidar-based 3d object detectors against adversarial attacks," *International Journal of Computer Vision*, vol. 132, no. 5, pp. 1592–1624, 2024.

[30] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[31] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[32] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 5–22, 2024.

[33] Y. Zhang, Y. Zhu, Z. Liu, C. Miao, F. Hajiaghajani, L. Su, and C. Qiao, "Towards backdoor attacks against lidar object detection in autonomous driving," in *ACM SenSys*, 2022, pp. 533–547.

[34] S. Li, Y. Wen, H. Wang, and X. Cheng, "BadLiDet: A simple backdoor attack against LiDAR object detection in autonomous driving," in *IEEE TrustCom*, 2023, pp. 99–108.

[35] S. Chaturvedi, G. Bagwe, L. Zhang, P. He, and X. Yuan, "MOBA: A material-oriented backdoor attack against LiDAR-based 3D object detection systems," in *AAAI*, 2026, arXiv:2511.09999.

[36] S. Chaturvedi, L. Zhang, W. Zhang, P. He, and X. Yuan, "BadFusion: 2D-oriented backdoor attacks against 3D object detection," in *IJCAI*, 2024.

[37] T. Chen, B. Du, J. Zhao, H. Xia, H. Wang, and J. Wang, "BadMDA: Towards backdoor injection during domain adaptation to collapse multi-agent perception," in *ACM MM*, 2025, pp. 975–983.

[38] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv preprint arXiv:1912.02771*, 2019.

[39] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 11 957–11 965, 2020.

[40] Y. Zeng, M. Pan, H. Just, L. Lyu, M. Qiu, and R. Jia, "Narcissus: A practical clean-label backdoor attack with limited information," in *ACM CCS*, 2023, pp. 771–785.

[41] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*, 2018.

[42] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[43] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[44] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *IEEE CVPR*, 2016, pp. 2574–2582.

[45] A. Shafahi, W. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *NeurIPS*, vol. 31, 2018.

[46] L. Fowl, M. Goldblum, P. yeh Chiang, J. Geiping, W. Czaja, and T. Goldstein, "Adversarial examples make strong poisons," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 339–30 351, 2021.

[47] W. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: Practical general-purpose clean-label data poisoning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 080–12 091, 2020.

[48] Z. Xiang, D. Miller, S. Chen, X. Li, and G. Kesidis, "A backdoor attack against 3d point cloud classifiers," in *IEEE/CVF ICCV*, 2021, pp. 7597–7607.

[49] X. Li, Z. Chen, Y. Zhao, Z. Tong, Y. Zhao, A. Lim, and J. Zhou, "Pointba: Towards backdoor attacks in 3d point cloud," in *IEEE/CVF ICCV*, 2021, pp. 16 492–16 501.

[50] Y. Bian, S. Tian, and X. Liu, "iba: Backdoor attack on 3d point cloud via reconstructing itself," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 7994–8008, 2024.

[51] K. Gao, J. Bai, B. Wu, M. Ya, and S.-T. Xia, "Imperceptible and robust backdoor attack in 3d point cloud," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1267–1282, 2023.

[52] G. Tian, W. Jiang, W. Liu, and Y. Mu, "Poisoning MorphNet for clean-label backdoor attack to point clouds," *arXiv preprint arXiv:2105.04839*, 2021.

[53] C. Wei, Y. Wang, K. Gao, S. Shao, Y. Li, Z. Wang, and Z. Qin, "PointNCBW: Toward dataset ownership verification for point clouds via negative clean-label backdoor watermark," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 191–206, 2025.

[54] N. Patel, P. Krishnamurthy, S. Garg, and F. Khorrami, "Bait and switch: Online training data poisoning of autonomous driving systems," *arXiv preprint arXiv:2011.04065*, 2020.

[55] X. Han, G. Xu, Y. Zhou, X. Yang, J. Li, and T. Zhang, "Physical backdoor attacks to lane detection systems in autonomous driving," in *ACM MM*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 2957–2968.

[56] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *IEEE/CVF CVPR*, 2020, pp. 11 040–11 048.

[57] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with Non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[58] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on Non-IID data," in *ICLR*, 2020.

[59] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7611–7623, 2020.

[60] T. Li, A. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *MLSys*, 2020, pp. 429–450.

[61] S. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *ICML*, 2020, pp. 5132–5143.

[62] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[63] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[64] C. Dinh, N. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

[65] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[66] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *ICML*, 2021, pp. 6357–6368.

[67] Y. Deng, M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, 2020.

[68] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[69] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.

[70] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik, "Lower bounds and optimal algorithms for personalized federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2304–2315, 2020.

[71] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[72] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[73] C. B. Issaid, A. Elgabli, J. Park, M. Bennis, and M. Debbah, "Communication-efficient decentralized learning over bipartite graphs," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 4150–4167, 2022.

[74] A. Elgabli, J. Park, S. Ahmed, and M. Bennis, "L-FGADMM: Layer-wise federated group ADMM for communication efficient decentralized deep learning," in *IEEE WCNC*, 2020, pp. 1–6.

[75] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1002–1012, 2020.

[76] P. Graf, J. Annoni, C. Bay, D. Biagioni, D. Sigler, M. Lunacek, and W. Jones, "Distributed reinforcement learning with ADMM-RL," in *ACC*, 2019, pp. 4159–4166.

[77] Z. Zheng, L. Song, Z. Han, G. Li, and H. Poor, "A stackelberg game approach to proactive caching in large-scale mobile edge networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5198–5211, 2018.

[78] X. Zhang, M. Khalili, and M. Liu, "Improving the privacy and accuracy of ADMM-based distributed algorithms," in *ICML*, 2018, pp. 5796–5805.

[79] C. Song, S. Yoon, and V. Pavlovic, "Fast ADMM algorithm for distributed optimization with adaptive penalty," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 753–759, 2016.

[80] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. Varshney, "Robust federated learning using admm in the presence of data falsifying byzantines," *arXiv preprint arXiv:1710.05241*, 2017.

[81] S. Yue, J. Ren, J. Xin, S. Lin, and J. Zhang, "Inexact-ADMM based federated meta-learning for fast and continual edge learning," in *ACM MobiHoc*, 2021, pp. 91–100.

[82] S. Zhou and G. Li, "Federated learning via inexact ADMM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 9699–9708, 2023.

[83] M. Ryu and K. Kim, "Differentially private federated learning via inexact ADMM," *arXiv preprint arXiv:2106.06127*, 2021.

[84] K. Wei, J. Li, M. Ding, C. Ma, H. Yang, F. Farokhi, S. Jin, T. Quek, and H. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[85] R. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[86] J. Konečnỳ, B. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[87] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *ICML*, 2017, pp. 1529–1538.

[88] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[89] Z. Parsons, F. Dou, H. Du, Z. Song, and J. Lu, "Mobilizing personalized federated learning in infrastructure-less and heterogeneous environments via random walk stochastic ADMM," *Advances in Neural Information Processing Systems*, vol. 36, pp. 36 726–36 737, 2023.

[90] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. Sayed, and W. Yin, "Walkman: A communication-efficient random-walk algorithm for decentralized optimization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2513–2528, 2020.

[91] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[92] C. Qi, H. Su, K. Mo, and L. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE CVPR*, 2017, pp. 652–660.

[93] C. Qi, L. Yi, H. Su, and L. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, vol. 30, 2017.

[94] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *IEEE CVPR*, 2018, pp. 4490–4499.

[95] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *IEEE/CVF CVPR*, 2019, pp. 770–779.

[96] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *IEEE/CVF ICCV*, 2019, pp. 1951–1960.

[97] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *IEEE/CVF CVPR*, 2018, pp. 7652–7660.

[98] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *IEEE/CVF CVPR*, 2020, pp. 10 529–10 538.

[99] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection," *International Journal of Computer Vision*, vol. 131, no. 2, pp. 531–551, 2023.

[100] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 8, pp. 2647–2664, 2021.

[101] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3D object detection from point cloud," in *IEEE/CVF CVPR*, 2020, pp. 11 873–11 882.

[102] W. Zheng, W. Tang, S. Chen, L. Jiang, and C.-W. Fu, "Cia-ssd: Confident iou-aware single-stage object detector from point cloud," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3555–3562, 2021.

[103] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel r-cnn: Towards high performance voxel-based 3d object detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 1201–1209, 2021.

[104] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "VoxelNeXt: Fully sparse VoxelNet for 3D object detection and tracking," in *IEEE/CVF CVPR*, 2023, pp. 21 674–21 683.

[105] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang, "Embracing single stride 3d object detector with sparse transformer," in *IEEE/CVF CVPR*, 2022, pp. 8458–8468.

[106] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3D object detection," in *IEEE/CVF ICCV*, 2021, pp. 3144–3153.

[107] H. Wang, C. Shi, S. Shi, M. Lei, S. Wang, D. He, B. Schiele, and L. Wang, "DSVT: Dynamic sparse voxel transformer with rotated sets," in *IEEE/CVF CVPR*, 2023, pp. 13 520–13 529.

[108] C. Xiang, C. Qi, and B. Li, "Generating 3D adversarial point clouds," in *IEEE/CVF CVPR*, 2019, pp. 9136–9144.

[109] T. A. Nguyen and A. T. Tran, "WaNet–imperceptible warping-based backdoor attack," in *ICLR*, 2021.

[110] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *IEEE/CVF ICCV*, 2021, pp. 16 463–16 472.

[111] T. A. Nguyen and A. T. Tran, "Input-aware dynamic backdoor attack," in *NeurIPS*, 2020.

[112] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV*, 2020, pp. 182–199.

[113] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNs by training set corruption without label poisoning," in *IEEE ICIP*, 2019, pp. 101–105.

[114] H. Souri, L. Fowl, R. Chellappa, M. Goldblum, and T. Goldstein, "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 165–19 178, 2022.

[115] J. Geiping, L. Fowl, R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," in *ICLR*, 2021.

[116] S. Hu, W. Liu, M. Li, Y. Zhang, X. Liu, X. Wang, L. Zhang, and J. Hou, "Pointcrt: Detecting backdoor in 3d point cloud via corruption robustness," in *ACM MM*, 2023, pp. 666–675.

[117] H. Liu, J. Jia, and N. Z. Gong, "PointGuard: Provably robust 3D point cloud classification," in *IEEE/CVF CVPR*, 2021, pp. 6186–6195.

[118] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *ACSAC*, 2019, pp. 113–125.

[119] S.-H. Chan, Y. Dong, J. Zhu, X. Zhang, and J. Zhou, "BadDet: Backdoor attacks on object detection," in *ECCV Workshops*, ser. LNCS, vol. 13801. Springer, 2023, pp. 396–412.

[120] Y. Cheng, W. Hu, and M. Cheng, "Attacking by aligning: Clean-label backdoor attacks on object detection," *arXiv preprint arXiv:2307.10487*, 2023.

[121] S. Ram, A. Nedić, and V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[122] X. Li and F. Orabona, "On the convergence of stochastic gradient descent with adaptive stepsizes," in *AISTATS*, 2019, pp. 983–992.

[123] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *IEEE CDC*, 2008, pp. 4177–4184.

[124] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5693–5700, 2019.

[125] Y. LeCun, C. Cortes, and C. Burges, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[126] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[127] Y. Wang, Y. Sun, Z. Liu, S. Sarma, M. Bronstein, and J. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[128] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu, "DUP-Net: Denoiser and upsampler network for 3D adversarial point clouds defense," in *IEEE/CVF ICCV*, 2019.

[129] Z. Wu, Y. Duan, H. Wang, Q. Fan, and L. Guibas, "IF-Defense: 3D adversarial point cloud defense via implicit function based restoration," *arXiv preprint arXiv:2010.05272*, 2020.

[130] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *USENIX Security Symposium*, 2020.

[131] Z. Hau, S. Demetriou, L. Muñoz-González, and E. C. Lupu, "Shadow-catcher: Looking into shadows to detect ghost objects in autonomous vehicle 3D sensing," in *European Symposium on Research in Computer Security (ESORICS)*, 2021.

[132] W. Lan, Y. Yang, H. Shen, and S. Li, "CloudFort: Enhancing robustness of 3D point cloud classification against backdoor attacks via spatial partitioning and ensemble prediction," *IET Computer Vision*, vol. 19, no. 1, p. e70047, 2025.

[133] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *AAAI SafeAI Workshop*, 2019, arXiv:1811.03728.

[134] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2018.

[135] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," in *NeurIPS*, 2021.

[136] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *RAID*, 2018.

[137] X. Qi, T. Xie, Y. Li, S. Mahloujifar, and P. Mittal, "Revisiting the assumption of latent separability for backdoor defenses," in *ICLR*, 2023.

[138] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE S&P*, 2019.

[139] R. Roriz, J. Cabral, and T. Gomes, "Automotive LiDAR technology: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 6282–6297, 2021.

[140] A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson, "A review of LIDAR radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28 099–28 128, 2015.

[141] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun, "Physically realizable adversarial examples for LiDAR object detection," in *IEEE/CVF CVPR*, 2020.

[142] J. Hu, T. Guo, and C. Han, "Stochastic linearized generalized alternating direction method of multipliers: Expected convergence rates and large deviation properties," *Mathematical Structures in Computer Science*, pp. 1–18, 2023.

[143] K. Chung and P. Erdős, "On the application of the Borel-Cantelli lemma," *Transactions of the American Mathematical Society*, vol. 72, no. 1, pp. 179–186, 1952.

[144] R. Rockafellar and R. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.