POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Adapt or Get Schooled: Two Methods of Adversarial Training for CAN Intrusion Detection Systems

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Stefano Viti**

Student ID: 247317
Advisor: Prof. Stefano Longari
Co-advisors: Michele Carminati
Academic Year: 2024-25

Abstract

As modern vehicles increasingly rely on data-driven detection mechanisms, ensuring their resilience to carefully crafted perturbations becomes critical for operational safety. This thesis investigates the trade-off between baseline detection performance and adversarial robustness in machine learning-based Intrusion Detection Systems (IDS) for Controller Area Network (CAN) environments.

The work establishes a comprehensive experimental framework grounded in clearly defined threat models, curated datasets, and reproducible evaluation procedures. Within this framework, two adversarial training strategies are studied: a self-adaptive adversarial training approach derived from the methodology proposed by Marchiori et al., and a novel teacher-guided adversarial training strategy that decouples adversarial example generation from the optimization of the target model.

Extensive experiments are carried out to assess detection performance and adversarial resistance in various configurations. The findings shed light on the ways in which various training paradigms affect robustness and show the circumstances in which adversarial training increases resilience without appreciably lowering baseline performance. Methodological issues for evaluating robustness in the CAN domain in addition to empirical evaluation are addressed, highlighting the significance of context-aware evaluation protocols. Lastly, directions for future work are described, including the exploration of complementary analysis techniques and broader threat scenarios.

Overall, this work advances our knowledge of adversarial robustness in automotive IDS and offers helpful advice for creating detection systems that are more resilient.

Keywords: cyberesecurity, controller area network, deep learning, adversarial samples, intrusion detection

Abstract in lingua italiana

Poiché i veicoli moderni si affidano sempre più a sistemi di rilevamento basati sui dati, garantire la loro resilienza a perturbazioni accuratamente studiate diventa fondamentale per la sicurezza operativa. Questa tesi indaga il compromesso tra prestazioni di rilevamento di base e robustezza avversariale nei sistemi di rilevamento delle intrusioni (IDS) basati sul machine learning per ambienti Controller Area Network (CAN).

Il lavoro stabilisce un quadro sperimentale completo basato su modelli di minaccia chiaramente definiti, dataset curati e procedure di valutazione riproducibili. All'interno di questo framework, vengono studiate due strategie di addestramento avversariale: un approccio di addestramento avversariale auto-adattivo derivato dalla metodologia proposta da Marchiori et al. e una nuova strategia di addestramento avversariale guidata da un insegnante che disaccoppia la generazione di esempi avversari dall'ottimizzazione del modello target.

Vengono condotti esperimenti approfonditi per valutare le prestazioni di rilevamento e la resistenza avversariale in varie configurazioni. I risultati fanno luce sui modi in cui i vari paradigmi di addestramento influenzano la robustezza e mostrano le circostanze in cui l'addestramento avversariale aumenta la resilienza senza ridurre sensibilmente le prestazioni di base. Vengono affrontate questioni metodologiche per la valutazione della robustezza nel dominio CAN, oltre alla valutazione empirica, evidenziando l'importanza di protocolli di valutazione che tengano conto del contesto. Infine, vengono descritte le direzioni per i lavori futuri, tra cui l'esplorazione di tecniche di analisi complementari e scenari di minaccia più ampi.

Nel complesso, questo lavoro amplia la nostra conoscenza della robustezza avversariale nei sistemi IDS per il settore automobilistico e offre utili consigli per la creazione di sistemi di rilevamento più resilienti.

Parole chiave: sicurezza informatica, controller area network, deep learning, esempi avversariali, rilevamento di intrusioni

Contents

Introduction

Modern vehicles rely on complex electronic control unit (ECU) networks that exchange safety-critical and comfort-related data on in-vehicle buses such as the Controller Area Network (CAN). Because CAN was designed for reliability and simplicity rather than security, a growing research and engineering effort aims to ensure that on-bus communications are free of malicious or malformed packets that could endanger vehicle functionality or passenger safety. A widely adopted defensive approach is the deployment of Intrusion Detection Systems (IDSs) that monitor bus traffic and flag (or block) messages that deviate from expected temporal, structural, or semantic patterns.

IDSs for CAN commonly address a set of conventional attack types that an adversary can mount on the bus, such as:

- Message Manipulation Attacks: attacks that manipulate in-vehicle communication semantics through the injection or replay of seemingly valid messages.
- Availability Attacks: attacks that are aimed at degrading the availability of the CAN bus or specific ECUs, potentially causing partial or complete denial of normal communication.
- Robustness-Oriented Attacks: attacks that target implementation weaknesses.

Early CAN IDSs used rule- or specification-based logic: handcrafted thresholds, message frequency checks, identifier whitelists and simple state machines that codified what was considered "normal". While deterministic and interpretable, these approaches have well-known limitations in scalability, maintainability, and their ability to generalize to subtle or evolving attack patterns.

To overcome those limits, research progressively integrated machine learning techniques into IDS design. Classical supervised and unsupervised algorithms, such as Support Vector Machines (SVMs), Random Forests, and clustering methods, were adopted to learn patterns from recorded traffic and detect anomalies that rules miss. More recently, deep learning models have been proposed to capture complex temporal and statistical dependencies in CAN data. These data-driven approaches have improved detection accuracy

and reduced manual tuning, but they also introduce new challenges: dependency on training data quality, computational cost, and potentially reduced interpretability.

As machine learning based IDSs became more capable, adversaries developed more sophisticated strategies specifically intended to defeat them. Attacks that exploit weaknesses in learned models, commonly referred to as adversarial attacks, are distinct from classical injection or replay in that they are explicitly designed to produce inputs that look benign to the detector while still achieving a malicious effect on the vehicle.

Adversarial attacks originated in domains such as computer vision and natural language processing, where machine learning models have been shown to be vulnerable to carefully crafted perturbations [1, 2]. In these domains, an attacker can make imperceptible changes to input images or text that lead a model to produce incorrect predictions, often with high confidence. The recognition of this vulnerability has prompted the development of numerous defence strategies.

While these methods can improve robustness, they often come at a significant cost: baseline performance on clean, unperturbed data typically decreases, computational complexity increases, and model interpretability may be reduced [3].

As research on adversarial robustness expands into the automotive domain, particularly the CAN protocol, these findings raise critical questions. Given the constrained and highly structured nature of CAN traffic, it is not immediately clear whether adversarial attacks represent a realistic threat, or whether implementing defensive strategies is practically worthwhile. The potential performance penalties and computational costs associated with adversarial defences, well-documented in other domains, must be carefully weighed against the actual benefits in the context of in-vehicle networks. Understanding this trade-off is essential to determine which defensive strategies, if any, could be effective and feasible for CAN-specific intrusion detection systems.

To address these questions, this thesis introduces a novel adversarial training strategy specifically designed for CAN-based intrusion detection systems, referred to as a *teacher-guided adversarial training* approach. The proposed method departs from conventional self-referential adversarial training schemes by decoupling the generation of adversarial examples from the model being trained. Instead, adversarial samples are produced by a separate reference model (the *teacher*) and are then used to train a target IDS (the *student*)

Alongside the proposed method, this thesis also revisits the adaptive online adversarial training approach proposed by Francesco Marchiori et al. in CANEDERLI [4]. While

CANEDERLI adversarial training represents an important step towards adversarially robust CAN IDSs, its original formulation exhibits several conceptual and practical issues that hinder its applicability in real-world deployments.

As part of this work, CANEDERLI training method is therefore re-engineered and corrected to address these limitations, resulting in an implementation that is both functionally sound and suitable for realistic evaluation.

By considering both a newly proposed teacher-guided strategy and a corrected version of an existing CAN-specific adversarial training method, this thesis provides a broader perspective on the design space of robustness-enhancing techniques for CAN intrusion detection systems. This dual contribution supports a more informed assessment of whether adversarial training is practically beneficial in constrained in-vehicle networks, and under which assumptions such technique can be effectively and realistically deployed.

To investigate, we conduct an extensive evaluation focusing primarily on baseline performance and adversarial robustness. Adversarial transferability is additionally considered to complement the analysis and to provide further insight into the interaction between different models and training methods under adversarial settings.

Baseline performance measures the model's accuracy and general behavior on clean, unperturbed CAN data. This serves as a reference point for understanding the trade-offs associated with robustness-enhancing techniques.

Adversarial robustness is assessed under two threat models:

- White-box robustness: we quantify how many adversarial examples can be generated from a given model itself. This indicates the model's vulnerability when the attacker has full knowledge of its architecture and parameters, and also allows relative comparisons between models: a model producing fewer successful adversarial examples is considered more robust.
- Grey-box robustness: we evaluate how well a model resists adversarial examples generated from other models trained on the same dataset. This captures the model's relative resilience to attacks crafted without full knowledge of its parameters, simulating a more realistic partial-knowledge scenario.

Adversarial transferability focuses on the ability of each model to generate effective attacks against other models. A model producing adversarial examples that successfully bypass the detection of many other models demonstrates higher transferability, revealing its potential as a strong attack generator.

This evaluation framework enables the assessment of the effectiveness of adversarial defense strategies when applied to deep learning–based CAN intrusion detection systems, with the goal of enhancing robustness against adversarial attacks while preserving satisfactory baseline performance.

The experimental analysis provides several insights into the role of adversarial training for CAN-based intrusion detection systems.

First, the results show that adversarial training can substantially improve robustness against crafted perturbations without necessarily compromising baseline detection performance, provided that training procedures are carefully designed. In particular, both adversarial training strategies considered in this work demonstrate the ability to reduce model vulnerability under white-box and grey-box attack scenarios.

Second, the study highlights that robustness gains are strongly influenced by architectural choices, dataset characteristics, and training configurations, emphasizing that adversarial robustness in the CAN domain cannot be considered a universal property but must be evaluated within a carefully defined experimental context.

Finally, the analysis of adversarial transferability shows that different training strategies affect not only how well a model defends against attacks, but also how effective it is at generating attacks against other models. This highlights that certain robustness-enhancing techniques may inadvertently produce stronger attack generators, an aspect that is particularly relevant when considering deployment in adversarial environments.

Overall, these findings contribute to clarifying when and how adversarial training can be effectively leveraged to enhance the security of machine learning–based intrusion detection systems in automotive networks.

Contributions The main contributions of this thesis can be summarized as follows:

- the proposal of a novel *teacher-guided adversarial training* strategy tailored to CAN-based intrusion detection systems, which leverages a surrogate model to guide the generation of informative adversarial samples and improve robustness without degrading baseline detection performance;
- the correction and re-implementation of the adaptive online adversarial training approach originally proposed in the CANEDERLI framework [4], enabling a faithful reproduction of the method, an extensive comparison with alternative training strategies, and the introduction of a second fully functional adversarial training pipeline within the same experimental framework;

- the design of an extensive evaluation framework for adversarial robustness, focusing on white-box and grey-box threat models, transferability across architectures, and the quantitative analysis of the trade-off between robustness to adversarial manipulation and detection performance on clean CAN traffic.

1 | Background

1.1. Key Terminology and Concepts Relevant to the CAN Protocol

The Controller Area Network (CAN) is a widely used in-vehicle communication protocol that allows multiple Electronic Control Units (ECUs) to exchange data reliably over a shared bus. Each CAN message, commonly referred to as a *data frame*, is composed of several fields that ensure reliable and deterministic data transmission.

A standard CAN frame consists of the following components:

- **Start of Frame (SOF)**: a single dominant bit indicating the beginning of a new message.
- **Arbitration Field**: contains the *Identifier* (11 bits in the standard format or 29 bits in the extended format) and the *Remote Transmission Request (RTR)* bit. The identifier determines both the message priority and its logical addressing within the network.
- **Control Field**: specifies the length of the data field via the *Data Length Code (DLC)*, which ranges from 0 to 8 bytes in classical CAN.
- **Data Field** (also referred as *payload*): carries the actual payload of the message, containing up to 8 bytes of application-specific data.
- **CRC Field**: includes a *Cyclic Redundancy Check* sequence and a delimiter bit, allowing receivers to verify message integrity.
- **ACK Field**: consists of an *Acknowledge Slot* and an *Acknowledge Delimiter*, through which receiving nodes confirm successful message reception.
- **End of Frame (EOF)**: composed of seven recessive bits marking the end of the frame.

CAN messages are broadcast over the network, and all nodes receive each frame simulta-

neously. They can be sent periodically, at fixed intervals, or event-triggered, depending on the ECU's function.

Arbitration is performed using a bitwise mechanism, ensuring that the message with the highest priority (i.e., the lowest identifier value) gains bus access without causing collisions. This deterministic arbitration process and the built-in error detection mechanisms make CAN highly reliable for real-time communication in safety-critical systems. The simplicity and lightweight nature of the protocol make it suitable for real-time automotive applications, but they also impose constraints that influence both attacks and defence strategies.

The constrained nature of CAN traffic (limited payload, absence of native authentication, and a shared bus architecture) means that traditional security mechanisms, such as encryption or complex verification procedures, are often impractical. As a result, detecting malicious or anomalous traffic requires monitoring patterns in message IDs, timing, and payload content rather than relying on cryptographic guarantees.

To address these challenges, Intrusion Detection Systems (IDSs) have been developed for CAN.

1.2. Intrusion Detection Systems definition

An *Intrusion Detection System* (IDS) is a fundamental security component designed to monitor, analyze, and detect unauthorized or malicious activities within computer systems or networks. The primary objective of an IDS is to identify abnormal behavior that may indicate potential security breaches, such as unauthorized access attempts, exploitation of vulnerabilities, or deviations from established communication patterns.

From a conceptual perspective, an IDS operates by continuously collecting data from various sources, such as network traffic, system logs, or application events, and comparing it against predefined rules or behavioral models. Depending on its detection methodology, an IDS can be broadly categorized into two main types:

- **Signature-Based IDS**: This approach relies on a database of known attack patterns or signatures. Each incoming event is matched against these signatures to detect known threats. Although effective against previously identified attacks, this method cannot detect novel or zero-day attacks.
- **Anomaly-Based IDS**: Instead of relying on known signatures, this approach models the normal behavior of the monitored system using statistical or machine learning

techniques. Any significant deviation from this normal profile is flagged as potentially malicious. While this method enables the detection of previously unseen attacks, it may suffer from higher false-positive rates.

In addition to these broad categories, IDSs can also be differentiated according to the underlying detection technique:

- **Rule-based or specification-based**: rely on manually defined thresholds and behavioral specifications, such as expected message frequencies, timing constraints, or allowed identifier ranges. These systems are typically lightweight and interpretable but may lack adaptability to dynamic environments.
- **Machine learning-based**: learn normal and abnormal patterns directly from observed traffic, often using classical models such as Support Vector Machines (SVMs), Decision Trees, or Random Forests. These approaches can generalize beyond explicitly defined rules but require a representative training dataset.
- **Deep learning-based**: capture complex temporal and statistical dependencies in the data using neural architectures such as Fully Connected Networks (FCNs), Convolutional Neural Networks (CNNs), or Long Short-Term Memory networks (LSTMs). These methods have shown high detection accuracy, especially in scenarios involving sequential or high-dimensional data, though at the cost of interpretability and computational complexity.

In terms of deployment, IDSs can be classified as either **Host-Based** (HIDS) or **Network-Based** (NIDS). A HIDS monitors the activity and integrity of a specific host by analyzing local logs, system calls, and file modifications, whereas a NIDS captures and inspects packets traversing the network to detect suspicious traffic patterns.

Modern IDS architectures often incorporate advanced data analysis techniques, such as feature extraction, dimensionality reduction, and ensemble learning, to enhance detection accuracy and reduce false alarms. Moreover, they may integrate with *Intrusion Prevention Systems* (IPS) to provide active defense capabilities, enabling the system to automatically block or mitigate detected attacks.

The effectiveness of an IDS is typically evaluated through metrics such as *Detection Rate*, *False Positive Rate*, and *Processing Latency*. In critical environments, such as industrial control systems, vehicular networks, or cloud infrastructures, achieving a balance between detection accuracy and computational efficiency remains a key research challenge. Consequently, ongoing research increasingly focuses on hybrid, adaptive, and context-aware IDS models capable of responding to evolving and sophisticated cyber threats.

1.3. Attacks on the CAN Protocol

In the context of in-vehicle networks, an *attack on the Controller Area Network (CAN)* refers to any malicious activity aimed at compromising the integrity, or availability of communications exchanged among Electronic Control Units (ECUs). Due to the broadcast nature of the CAN bus and the absence of built-in security mechanisms such as authentication or encryption, the protocol is inherently vulnerable to a wide range of cyberattacks.

The CAN protocol was originally designed for reliability and fault tolerance in real-time distributed control systems, rather than for resilience against adversarial behavior. As a result, any node connected to the bus can transmit arbitrary messages that are accepted by all other nodes, provided they comply with the protocol's timing and frame format. This characteristic allows attackers with physical or logical access to the bus to inject, modify, or replay messages, potentially manipulating vehicle functionalities or disrupting communication flows.

Typical attacks against the CAN protocol can be broadly categorized based on their primary objective and attack mechanism. While these categories are not mutually exclusive, they provide a structured view of the most common threat classes.

- **Message Manipulation Attacks**: these attacks aim to alter the semantic meaning of in-vehicle communication by injecting or reusing valid-looking messages.
  - *Injection attacks*: the adversary transmits fabricated CAN frames with valid identifiers and data fields to impersonate legitimate ECUs or trigger unintended actions. For instance, spoofing braking or steering commands may lead to severe safety consequences.
  - *Replay attacks*: previously captured legitimate CAN messages are resent at a later time to reproduce valid system behavior in an inappropriate or malicious context, potentially bypassing simple detection mechanisms.
- **Availability Attacks**: these attacks target the availability of the CAN bus or specific ECUs, resulting in partial or complete denial of normal communication.
  - *Denial-of-Service (DoS) attacks*: by continuously transmitting high-priority messages, the attacker exploits CAN's arbitration mechanism to monopolize the bus and prevent legitimate traffic from being delivered.
  - *Bus-off attacks*: by deliberately inducing transmission errors, an attacker can force one or more ECUs into the *bus-off* state, effectively isolating them from

the network.

- **Robustness-Oriented Attacks**: these attacks aim to expose implementation weaknesses rather than directly achieving a specific functional outcome.
  - *Fuzzing or flooding attacks*: random, malformed, or high-rate CAN frames are injected to trigger unexpected behaviors, software faults, or crashes in ECUs, often serving as a preliminary step for more targeted attacks.

While many of these attacks require physical access to the in-vehicle network through diagnostic ports such as the OBD-II interface, the increasing connectivity of modern vehicles (via wireless interfaces, infotainment systems, or telematics units) has expanded the attack surface, enabling remote exploitation scenarios. Consequently, the detection and mitigation of CAN-based attacks have become critical components of automotive cybersecurity research, motivating the development of lightweight and real-time Intrusion Detection Systems specifically tailored for CAN environments.

1.4. Definition of Adversarial Attacks and Techniques to Generate Perturbation

Adversarial evasion attacks are carefully crafted inputs intended to mislead a trained machine learning model while remaining close to legitimate inputs [1, 2]. Formally, given a model f and an input x with correct label $y = f(x)$, an adversarial example x' is generated such that $f(x') \neq y$ while x' is "close" to x according to a chosen norm or similarity measure.

Adversarial attacks can be categorized based on the attacker's knowledge of the target model:

- **White-box attacks:** the attacker has full access to the model architecture, its parameters, and its training dataset.
- **Grey-box attacks:** the attacker has partial knowledge, for example access to the dataset used to train the model or access to some model internals.
- **Black-box attacks:** the attacker can only query the model and observe outputs, without access to its internal structure or parameters.

White-box attacks typically achieve higher success rates in practice, but they represent a less realistic threat model for many real-world deployments. Grey-box and black-box attacks are often implemented using a *substitute model* (also referred as *oracle*): an at-

tacker trains or acquires a surrogate model and generates adversarial inputs against that surrogate. Successful adversarial examples are then transferred to the target model. This procedure relies on the so-called *transferability property*: adversarial inputs that deceive one model often have a substantial probability of deceiving other models. Transferability is commonly attributed to the fact that, despite differences in architecture or training details, models trained on the same task tend to learn similar decision functions.

Common families of techniques for generating adversarial perturbations include the following:

Gradient-based methods These algorithms leverage gradients of the model's loss with respect to the input to compute efficient perturbations.

- **Fast Gradient Sign Method (FGSM)** [2]: a single-step method that perturbs the input in the direction of the gradient sign:

$$x' = x + \varepsilon \cdot \operatorname{sign}\big(\nabla_x \mathcal{L}(f(x), y)\big),$$

  where ε controls the perturbation magnitude. FGSM is computationally cheap and commonly used as a baseline.

- **Basic Iterative Method (BIM)** [5] and **Projected Gradient Descent (PGD)** [6]: iterative multi-step extensions of FGSM that apply small gradient steps while keeping the perturbed input within an allowed L_p-ball. Formally, they can be written as

$$x^{(t+1)} = \Pi_{\mathcal{B}_p(x,\varepsilon)}\Big(x^{(t)} + \alpha \cdot \operatorname{sign}\big(\nabla_x \mathcal{L}(f(x^{(t)}), y_t)\big)\Big),$$

  where $\Pi_{\mathcal{B}_p(x,\varepsilon)}$ denotes the projection onto the L_p-ball of radius ε around the original input x, and α is the step size. The main difference between the two methods lies in the initialization: BIM starts from the original input ($x^{(0)} = x$), while PGD typically uses a random initialization within the allowed perturbation set ($x^{(0)} = x + \delta^{(0)},\ \delta^{(0)} \sim \mathcal{U}(-\varepsilon, \varepsilon)^d$). PGD is widely regarded as a strong first-order adversary due to this random start and multi-step optimization.

- **Targeted vs. Untargeted variants**: gradient-based attacks can be targeted (force model to predict a specific y_t) or untargeted (cause any misclassification). The loss $\mathcal{L}$ is chosen accordingly.

- **Momentum and adaptive step variants**: methods such as Momentum Iterative FGSM (MI-FGSM [7]) add a momentum term to stabilize updates and improve transferability across models.

Optimization-based methods These treat adversarial example generation as a constrained optimization problem and often use continuous optimization techniques to find minimal perturbations.

- **DeepFool** [8]: iteratively approximates the classifier boundary as linear and computes the smallest perturbation that crosses this boundary, producing small-norm adversaries. The method repeats linearization and projection steps until the decision changes.
- **Jacobian-based Saliency Map Attack (JSMA)** [9]: crafts sparse perturbations by computing input saliency and perturbing features that most influence the target output.

When adversarial techniques are applied to temporal or protocol data (e.g., IDS inputs or CAN message streams), additional constraints and considerations arise:

- **Discrete and bounded feature spaces**: CAN payload bytes and message identifiers are discrete (0–255 for bytes, fixed bit-length identifiers). Perturbations must preserve syntactic validity; thus continuous gradient steps require quantization or integer-constrained optimization.
- **Timing and real-time constraints**: on a CAN bus, inter-frame timing and message periodicity are critical. Effective adversarial perturbations can include timing modifications (e.g., altering message frequency) which must respect real-time feasibility.
- **Semantic plausibility (stealthiness)**: perturbations should avoid producing obviously invalid or out-of-range signals (e.g., impossible sensor values) to evade downstream sanity checks or specification-based detectors.
- **Payload-DLC consistency**: the DLC value reported in the data frame must be consistent with the length of the payload.

1.5. Machine Learning and Deep Learning Models

Machine learning (ML) is a branch of artificial intelligence that focuses on designing algorithms capable of learning patterns and relationships from data without being explicitly programmed. Instead of relying on manually defined rules, ML models infer decision boundaries or predictive functions by generalizing from examples. This data-driven approach has proven highly effective in numerous domains, including computer vision, natural language processing, and cybersecurity, where patterns of interest are often complex,

nonlinear, and high-dimensional.

At its core, a machine learning model learns a mapping function $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$, parameterized by a set of learnable parameters θ, that minimizes a loss function $\mathcal{L}(f_\theta(x), y)$ over a training dataset composed of input-output pairs $\{(x_i, y_i)\}_{i=1}^N$. Depending on the learning paradigm, ML methods can be categorized as:

- **Supervised learning**, where the model is trained using labeled data, i.e., each input x_i is associated with a ground truth label y_i. The objective is to learn to predict y for unseen samples. Examples include classification and regression tasks.
- **Unsupervised learning**, where the model is trained on unlabeled data to discover hidden structures or patterns, such as clusters or latent representations. Techniques like clustering or autoencoders fall into this category.
- **Semi-supervised learning**, which combines labeled and unlabeled data to leverage partially annotated datasets.
- **Reinforcement learning**, where an agent learns optimal actions through interaction with an environment, receiving feedback in the form of rewards or penalties.

In cybersecurity applications, such as Intrusion Detection Systems (IDSs), supervised and unsupervised learning are the most common paradigms. Supervised models are used when labeled attack data are available, while unsupervised or anomaly-based approaches are preferred when attack patterns are unknown or rare.

From Machine Learning to Deep Learning

Traditional ML algorithms, such as Support Vector Machines (SVMs), Random Forests, and k-Nearest Neighbors (k-NN), rely heavily on manually engineered features to represent input data. The effectiveness of these models is therefore constrained by the quality and expressiveness of the chosen feature set.

Deep learning (DL), a subfield of machine learning, overcomes this limitation by automatically learning hierarchical feature representations directly from raw data. Deep learning models consist of multiple processing layers, each composed of neurons that perform nonlinear transformations, which allow the model to capture complex abstractions and latent structures. This hierarchical representation learning is particularly advantageous in domains where data exhibit spatial, temporal, or statistical dependencies, such as CAN bus traffic or network communication patterns.

Formally, a deep neural network implements a composite function:

$$f_\theta(x) = f^{(L)}(f^{(L-1)}(\dots f^{(1)}(x))),$$

where each layer $f^{(l)}$ transforms the output of the previous layer through a learned affine transformation followed by a nonlinear activation. The model parameters θ are optimized by minimizing a loss function using gradient-based optimization methods such as stochastic gradient descent (SGD) or Adam.

Deep learning models have shown outstanding performance across many application areas due to their ability to generalize from large-scale data, automatically extract discriminative features, and model nonlinear and temporal dependencies. The following subsections provide a detailed description of three representative deep architectures widely adopted in cybersecurity and intrusion detection research: Fully Connected Networks (FCNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) networks.

1.5.1. Fully Connected Networks

A *Fully Connected Network* (FCN), also known as a *Multilayer Perceptron* (MLP), is the most fundamental form of deep neural network. In an FCN, each neuron in one layer is connected to every neuron in the subsequent layer, forming a dense connectivity pattern.

Given an input vector $x \in \mathbb{R}^n$, each hidden layer performs the transformation:

$$h^{(l)} = \sigma\big(W^{(l)}h^{(l-1)} + b^{(l)}\big),$$

where $W^{(l)}$ and $b^{(l)}$ denote the weight matrix and bias vector of layer l, respectively, and $\sigma(\cdot)$ is a nonlinear activation function such as ReLU, tanh, or sigmoid. The final output layer applies a suitable activation (e.g., softmax for classification).

FCNs are universal function approximators and can learn arbitrary mappings between input and output spaces. However, they lack explicit mechanisms to capture spatial or temporal dependencies, which limits their performance on structured data such as images or time series. In IDS applications, FCNs are often employed as baseline models or as components in hybrid architectures due to their simplicity and interpretability.

1.5.2. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are designed to exploit spatial locality and translation invariance in structured data. Originally developed for image recognition tasks,

CNNs have also been successfully applied to one-dimensional temporal or sequential data, including network traffic and CAN bus messages.

A CNN layer applies a set of learnable convolutional filters (kernels) that slide across the input to produce feature maps:

$$h_k^{(l)} = \sigma\big((x * W_k^{(l)}) + b_k^{(l)}\big),$$

where $*$ denotes the convolution operation, $W_k^{(l)}$ the k-th kernel, and $b_k^{(l)}$ its bias term. Convolutional layers are typically followed by pooling operations (e.g., max-pooling or average-pooling) that reduce dimensionality and promote translational invariance.

By stacking multiple convolutional and pooling layers, CNNs learn hierarchical representations: early layers capture low-level patterns (e.g., frequency or local correlations), while deeper layers encode higher-level abstract features. This makes CNNs particularly suitable for detecting spatial or local temporal anomalies in IDS tasks, where correlations between adjacent features or message bytes carry important semantic information.

1.5.3. Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks are a special type of Recurrent Neural Network (RNN) specifically designed to capture long-range temporal dependencies in sequential data. Traditional RNNs suffer from the vanishing or exploding gradient problem, which limits their ability to learn dependencies over long sequences. LSTMs address this limitation through the introduction of memory cells and gating mechanisms that regulate information flow.

An LSTM cell maintains a cell state c_t and a hidden state h_t that evolve over time as follows:

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f),\\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i),\\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o),\\
\tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c),\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t,\\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}$$

where f_t, i_t, and o_t represent the forget, input, and output gates respectively, and $\odot$ denotes element-wise multiplication. These gates dynamically control which information is retained, updated, or discarded at each time step.

LSTMs are highly effective for modeling temporal correlations, making them suitable for analyzing time-dependent data such as CAN traffic, where message sequences and inter-frame timings carry semantic and behavioral patterns.

2 | Motivation

2.1. Problem Statement

The increasing adoption of machine learning (ML) techniques for in-vehicle Intrusion Detection Systems (IDSs) has substantially improved the ability to detect malicious activities on the Controller Area Network (CAN) bus. However, as ML models become more prevalent in safety-critical automotive contexts, they also inherit vulnerabilities extensively documented in other domains, most notably, their susceptibility to adversarial attacks. Adversarial attacks are carefully crafted perturbations to legitimate inputs that induce incorrect model outputs; in the IDS setting this typically translates into malicious traffic being misclassified as benign.

While adversarial machine learning is a mature research area in computer vision and general network security, its application to CAN-based IDSs raises domain-specific questions. The CAN protocol is characterized by strict constraints on message format, payload size, identifiers, and timing. These constraints may (a) limit the attacker's ability to craft perturbations that are both effective and physically plausible, and (b) influence the transferability and detectability of adversarial examples across different IDS paradigms. A rigorous investigation is therefore required to assess whether adversarial training, widely adopted in other domains, can be effectively adapted to CAN-based IDSs, and to quantify the resulting trade-offs in terms of detection performance, and robustness gains.

2.2. State of the Art

Early intrusion detection systems for the Controller Area Network relied primarily on rule-based or specification-based approaches. These systems employed handcrafted logic, such as fixed thresholds on message frequency, identifier whitelists, and finite-state models describing expected communication patterns. While deterministic and easily interpretable, such approaches suffer from limited scalability, high maintenance cost, and poor generalization to subtle, evolving, or previously unseen attack strategies.

To address these limitations, subsequent research introduced machine learning techniques into CAN IDS design. Classical supervised and unsupervised models, including Support Vector Machines (SVMs), Random Forests, and clustering-based anomaly detectors, enabled data-driven learning of traffic patterns and improved detection of deviations that static rules fail to capture.

The adoption of deep learning marked a significant shift in CAN intrusion detection research. Early foundational works appeared around 2016 and laid the groundwork for many subsequent studies. Notable examples include the work by Kang et al. [10], who proposed a feed-forward deep neural network for in-vehicle network security, and Taylor et al. [11], who demonstrated the effectiveness of Long Short-Term Memory (LSTM) networks for anomaly detection in CAN data by explicitly modeling temporal sequences. In the same period, the technical report *Detecting Attacks on the CAN Protocol With Machine Learning* [12] from the University of Michigan further highlighted the potential of learning-based approaches for CAN security.

In the following years, an increasing variety of deep learning architectures have been adapted to CAN-based IDSs. Convolutional Neural Networks (CNNs) have been employed to capture local temporal and structural patterns in CAN message sequences, as demonstrated by Song et al. [13]. Generative and representation-learning approaches have also been explored, including GAN-inspired frameworks such as GIDS proposed by Seo et al. [14], which model benign traffic distributions to detect anomalies. Later works explored unsupervised and semi-supervised deep learning systems, such as CANet by Hanselmann et al. [15] and CANdito by Longari et al. [16], aimed to reduce reliance on labeled attack data.

As learning-based CAN intrusion detection systems have become increasingly effective, the threat landscape has evolved accordingly. Rather than relying solely on classical message injection or replay techniques, attackers can deliberately exploit vulnerabilities in the learned decision boundaries of ML-based detectors. These strategies, commonly referred to as adversarial attacks, aim to induce misclassification by crafting inputs that remain functionally malicious for the vehicle while appearing benign to the IDS.

In recent years, the impact of adversarial manipulation in CAN-based intrusion detection systems has been explicitly investigated in the literature. Longari et al. analyzed the robustness of automotive IDSs against evasion attacks in [17], studying how carefully crafted perturbations influence detection performance under realistic constraints. This line of work was further extended in [18], where multiple CAN-based IDS models and adaptations of known adversarial strategies were evaluated under different knowledge

assumptions, revealing structural vulnerabilities that are not observable through standard performance metrics alone.

Complementary to these analyses, Aloraini et al. [19] proposed an adversarial attack methodology specifically designed for CAN-based intrusion detection systems in connected and autonomous vehicles. Their work demonstrates how adversarially crafted CAN messages can effectively evade learning-based IDSs, highlighting the susceptibility of such systems to targeted perturbations and reinforcing the need for robustness-oriented evaluation and defense mechanisms.

A wide variety of adversarial defence strategies have been proposed in the broader machine learning literature, aiming to increase model robustness against adversarial perturbations. Among the most common are:

- **Adversarial training**, where models are exposed to adversarial examples during training to improve robustness. It is effective against many first-order attacks but typically increases training cost and may degrade clean-data accuracy.
- **Input preprocessing and feature smoothing**, such as quantization, noise filtering, and transformation-based defenses intended to remove adversarial perturbations before classification.
- **Gradient obfuscation and regularization**, which attempt to reduce the usefulness of gradient information for attackers, albeit often providing only illusory or temporary robustness.
- **Detection-based methods**, which seek to identify adversarial inputs using auxiliary detectors, ensemble disagreement, statistical tests, or distributional checks.

These techniques have been adapted in preliminary ways to CAN IDSs. Example efforts include the use of Dynamic Label Watermark in GRU-based IDSs [20] and generative-model-based frameworks (e.g., GAN-inspired systems) to model benign traffic distributions and detect anomalies [14, 21]. It is worth noting, however, that these generative approaches were not explicitly designed to address adversarial robustness; while modeling the distribution of benign traffic may implicitly affect the detector's sensitivity to perturbed inputs, their resilience against adversarial attacks has generally not been the primary design objective nor systematically evaluated.

Overall, the literature remains fragmented. Important gaps include systematic evaluations of different adversarial training strategies, their effectiveness against adversarial attacks under realistic threat models, and assessments conducted on representative CAN datasets.

2.3. Goals and Contributions

The primary goal of this work is to investigate whether adversarial training strategies can be effectively applied to deep learning-based CAN intrusion detection systems, improving robustness against adversarial manipulation while preserving reliable detection performance on clean CAN traffic. In the automotive domain, where false alarms and missed detections may have safety-critical consequences, any gain in robustness must be carefully balanced against potential degradations in baseline accuracy.

To address this challenge, this thesis pursues the following objectives:

- to design and evaluate a novel *teacher-guided adversarial training* strategy tailored to the structural and semantic constraints of the CAN protocol;
- to revisit, correct, and re-implement a state-of-the-art adaptive online adversarial training approach proposed in the literature, enabling a fair and reproducible comparison with alternative training strategies;
- to systematically compare standard and adversarially trained models in terms of detection performance on clean traffic, robustness to adversarial attacks, and adversarial transferability;
- to define a structured evaluation framework for adversarial robustness in CAN-based IDSs, relying on domain-relevant metrics and realistic white-box and grey-box threat models.

Building on these goals, the contributions of this thesis consist of an extensive experimental analysis of adversarial training techniques for CAN-based intrusion detection systems, highlighting their practical benefits and limitations. In particular, this work provides empirical evidence on the trade-offs between robustness and clean-data performance, assesses the transferability of adversarial vulnerabilities across different model architectures, and offers practical insights and guidelines for the integration of adversarial training into automotive IDS pipelines.

2.4. Challenges and Practical Constraints

Pursuing the stated goals requires addressing several domain-specific challenges that influence both the design of adversarial defenses and the interpretation of experimental results:

- **Protocol constraints and semantic validity:** CAN payloads and identifiers are

discrete and often semantically meaningful. Adversarial perturbations must preserve syntactic correctness and, ideally, semantic plausibility to avoid trivial detection by rule-based sanity checks or protocol-level inconsistencies. These constraints limit the space of admissible perturbations and complicate the direct application of adversarial methods originally designed for continuous domains.

- **Dataset availability and generalizability:** Publicly available, well-labeled CAN datasets are limited and may not fully capture the diversity of real-world vehicles, ECUs, and driving conditions. Ensuring that robustness gains generalize across different vehicle platforms and traffic characteristics remains a non-trivial challenge and motivates evaluations across multiple datasets and architectures.
- **Computational cost and scalability:** Adversarial training substantially increases training complexity by requiring repeated gradient-based perturbation generation during optimization. While this work does not focus on runtime benchmarking, the additional computational burden introduces a practical trade-off between robustness, training scalability, and experimental coverage. This constraint influences the feasible number of architectures, datasets, and attack configurations that can be explored while maintaining a controlled and interpretable experimental setup.

Addressing these challenges is essential to design IDSs that balance robustness, accuracy, and feasibility in real-world automotive contexts. Addressing these challenges is essential for designing intrusion detection systems that balance robustness, accuracy, and feasibility in real-world automotive contexts. Moreover, these constraints explicitly guide the methodological and experimental choices adopted throughout this thesis.

2.5. Scope

This work focuses on adversarial interactions between attackers and ML-based CAN IDSs under clearly specified threat models. Results will be contextualized with respect to the chosen datasets and threat assumptions. Limitations will be explicitly stated to guide future work.

3 | Approach

This study examines adversarial interactions between attackers and machine-learning-based CAN intrusion detection systems under explicitly defined threat models. All results are interpreted in the context of the selected datasets and threat assumptions, and the corresponding limitations are clearly stated to inform future research directions.

3.1. Approach Overview

This chapter presents the adversarial training strategies investigated in the study and provides a high-level overview of the proposed methodology.

The objective is to improve the robustness of CAN-based intrusion detection systems against adversarial manipulation, while preserving reliable detection performance on clean CAN traffic. At a conceptual level, the approach builds upon the idea that exposing an IDS to carefully crafted adversarial perturbations during training can lead to more stable decision boundaries and improved generalization. However, in the context of CAN traffic, adversarial training must respect strict semantic and protocol constraints: perturbations must remain physically realizable and preserve the temporal structure of messages.

To address these challenges, this work considers and extends two complementary adversarial training paradigms:

- a novel *teacher-guided adversarial training* strategy, introduced in this thesis;
- an *self-adaptive online adversarial training* strategy, derived from the CANEDERLI framework [4].

Both approaches integrate adversarial example generation directly into the training process, but differ fundamentally in how adversarial samples are produced and how they influence model learning.

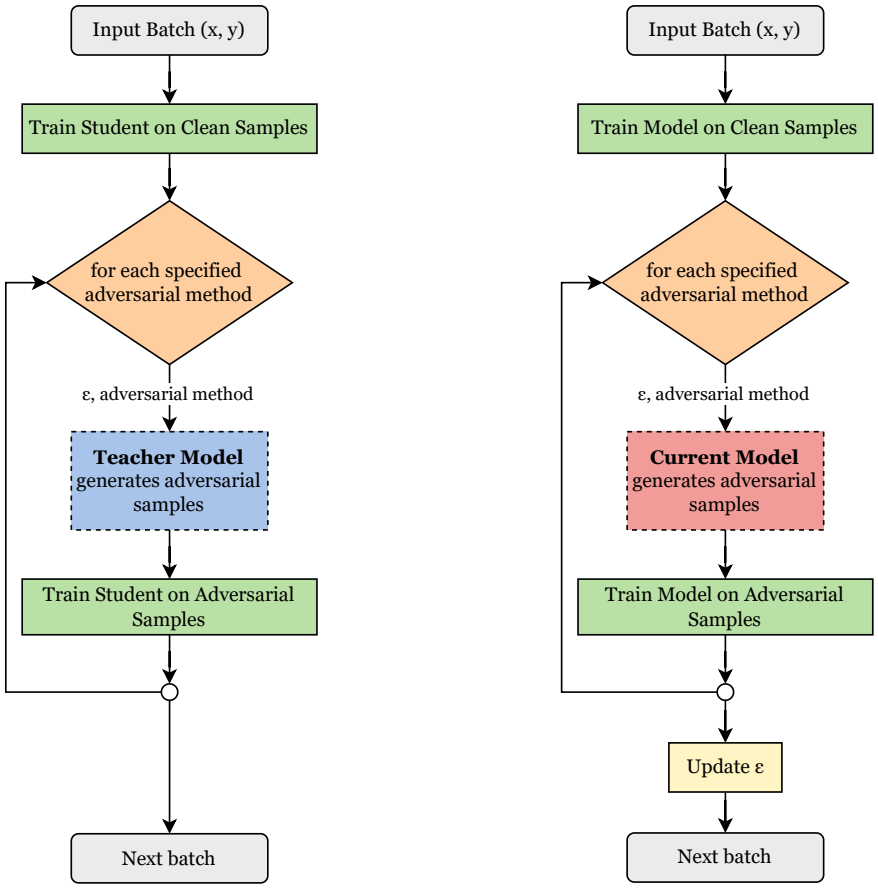
The teacher-guided adversarial training strategy proposed in this thesis decouples adversarial example generation from the optimization of the target model. Instead of relying on the model under training to generate adversarial perturbations, a separate *teacher model*

is introduced to guide the adversarial process.

At a high level, this strategy can be interpreted as a form of *informed adversarial regularization*: the teacher encodes prior knowledge about the data distribution and attack surface, while the student leverages this information to learn more robust and smoother decision boundaries.

In contrast, the adaptive implementation operates through what is known in the literature as a form of *self-play*, in which the model repeatedly attacks itself and learns from its own weaknesses.

The two strategies are conceptually summarized in Figure 3.1.



(a) Teacher-guided pipeline (b) Self-adaptive pipeline

Figure 3.1: Comparison of adversarial training implementation pipelines

3.2. Teacher-Guided Adversarial Training

The core contribution of this work is a teacher-guided adversarial training strategy specifically designed for CAN-based intrusion detection systems. The key idea is to leverage a separate, pre-trained reference model (the *teacher*) to generate adversarial examples that are subsequently used to guide the training of a target IDS (the *student*). Unlike conventional adversarial training approaches, where adversarial samples are generated using the same model that is being trained, the proposed strategy decouples adversarial example generation from the student model, aiming to improve robustness in a more controlled and meaningful way.

From a conceptual standpoint, the proposed approach can be interpreted as a form of informed adversarial regularization. Rather than exposing the student to adversarial perturbations generated with respect to its own evolving parameters, the teacher provides a stable adversarial reference that encodes prior knowledge about the data distribution and the attack surface. This allows the student to focus on learning robust and smoother decision boundaries guided by a fixed reference, instead of continuously adapting to a moving adversarial objective.

In this framework, the teacher model is trained using standard (non-adversarial) procedures and then used as a fixed surrogate to craft adversarial samples. These adversarial examples are generated to evade the teacher's decision boundary, but are injected into the training process of a distinct student model, which learns to correctly classify both clean and adversarial inputs.

This separation yields two key conceptual advantages:

- First, it stabilizes adversarial training by avoiding the tight feedback loop present in fully online approaches, in which adversarial examples are generated with respect to a rapidly evolving decision boundary. By relying on a fixed teacher, adversarial perturbations remain consistent across training iterations, reducing optimization oscillations.
- Second, it allows the student to benefit from informative adversarial signals even when its own decision boundary is still underdeveloped, particularly during the early stages of training.

In contrast, conventional adversarial training tightly couples adversarial example generation and model optimization, often resulting in unstable training dynamics, especially in highly non-convex settings such as deep neural networks for intrusion detection. In

such scenarios, adversarial samples generated early in training may be poorly aligned with meaningful attack patterns, while later adversarial samples may become excessively specialized to the current model state. The teacher-guided strategy explicitly mitigates these effects by decoupling robustness supervision from student optimization.

At the same time, this design introduces an explicit dependency between teacher quality and student robustness. When the teacher captures meaningful, attack-relevant patterns, the adversarial samples it generates provide effective training signals for the student. Conversely, if the teacher fails to generalize, the resulting guidance may be of limited utility or even detrimental. The implications of this dependency are explicitly analyzed in the experimental evaluation (Chapter 5).

Both the teacher and the student are binary classifiers tasked with distinguishing between benign and malicious CAN messages. Each supervised model receives as input the feature vector

$$\{\texttt{ID},\ \Delta t,\ \texttt{byte}_1, \ldots, \texttt{byte}_8\},$$

where $\texttt{byte}_i$ denotes the eight payload bytes of the CAN frame, normalized to the interval $[0, 1]$, and Δt is the inter-arrival time relative to the previous message with the same `ID`. This feature representation follows common practice in the CAN intrusion-detection literature and facilitates the construction of adversarial perturbations that remain semantically consistent with protocol constraints.

The training process is structured in two distinct phases: teacher training and student training.

Teacher training. In the first phase, the teacher model is trained using standard (non-adversarial) training on the original CAN dataset. This dataset contains both benign and malicious samples and reflects the baseline data distribution observed by a CAN IDS.

Using the binary cross-entropy (BCE) as the loss function, the teacher is trained until convergence and then frozen; its parameters remain fixed during the subsequent student training phase. The role of the teacher is not to be deployed as the final IDS, but to act as a stable and expressive adversarial example generator.

Student training with teacher guidance. Once teacher training is completed, the student model is trained using a structured input procedure that combines clean and adversarially perturbed samples. Training is performed in an online fashion, with adversarial samples generated dynamically during each training iteration using the frozen teacher model.

The student training pipeline supports configurable class balancing strategies. Depending on the experimental configuration, class imbalance can be addressed through undersampling of the majority class, oversampling of the minority class, or the use of class-weighted loss functions. The selected class balancing strategy is treated as a hyperparameter and remains fixed for each experiment. Additional hyperparameters include the set of adversarial attack methods used during training and the maximum perturbation budget ε.

During training, for each sample drawn from the original dataset, the student is presented with multiple inputs derived from the same original message. First, the original (clean) sample is provided to the student. Then, one or more adversarial versions of the same sample are generated using the teacher model, each obtained by applying a different adversarial attack method within the predefined perturbation budget (see Figure 3.1a). When the original sample is malicious, these perturbations produce non-trivial adversarial examples designed to evade detection. When the original sample is benign, no meaningful adversarial perturbation exists and all perturbed versions coincide with the original input.

As a consequence of this training procedure, benign samples are intentionally observed multiple times within the same training epoch. This repetition is a deliberate design choice aimed at preserving the selected class balancing strategy.

Overall, during training the student observes:

- original benign samples, potentially repeated across the same epoch;
- original malicious samples;
- multiple adversarially perturbed versions of malicious samples generated by the teacher model using different attack methods.

The resulting method can be formalized as a joint optimization over clean samples and their teacher-generated adversarial counterparts, explicitly balancing clean and adversarial contributions within each training step.

Formally, the loss function of the student model is:

$$\mathcal{L}_{student} = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\frac{1}{1+|\mathcal{A}|} \left(\mathcal{L}(f_S(x), y) + \sum_{a\in\mathcal{A}} \mathcal{L}(f_S(x'_{teacher,a}), y) \right) \right]$$

where $\mathcal{A}$ is the set of adversarial attacks, $x'_{teacher,a}$ denotes an adversarial example generated by applying attack $a \in \mathcal{A}$ to the *fixed* teacher model, and $\mathcal{L}$ is the binary cross-entropy (BCE) loss.

3.3. Correction and Re-Implementation of CANEDERLI Adversarial Training

This thesis re-implements and extends the adversarial training pipeline originally introduced in Marchiori et al. [4]. The study proposes an adaptive online adversarial training strategy tailored to CAN-based intrusion detection systems.

At a conceptual level, the CANEDERLI approach is based on *online, self-adaptive adversarial training*. During training, the model under optimization simultaneously plays the role of both defender and adversary: it is used to generate adversarial perturbations against itself, which are then immediately incorporated into the training process.

At each training step, the current state of the model is exploited to craft adversarial examples from malicious samples within the current mini-batch. These adversarial samples are generated using gradient-based evasion attacks under a constrained perturbation budget intended to preserve CAN payload semantics. The resulting adversarial inputs are then mixed with clean samples and used to update the model parameters.

This closed-loop interaction creates a continuously evolving adversarial landscape, in which the model is forced to adapt to perturbations specifically tailored to its most recent decision boundaries. The intended effect is to progressively harden the IDS against evasion attacks that exploit its current vulnerabilities.

While this strategy promotes robustness, it also tightly couples adversarial example generation to the instantaneous state of the model. In practice, this coupling may amplify training instability or introduce bias when the model has not yet learned a meaningful decision boundary, particularly during early training stages.

Despite its relevance and its contribution toward adversarially robust CAN IDSs, the original CANEDERLI implementation exhibits several implementation-level issues that hinder reliable reproduction and limit applicability in realistic deployment scenarios.

To address these limitations, we introduce a series of targeted corrections and refinements along three orthogonal dimensions: (i) the supervised learning formulation, (ii) the input feature representation, and (iii) the procedure used to generate and integrate adversarial examples during training. The main modifications and their motivations are detailed below.

Problem formulation and input representation The original CANEDERLI study formulates intrusion detection as a *multiclass* classification problem operating on single-

frame inputs represented by the tuple:

$$\{\texttt{ID}, \texttt{DLC}, \Delta t, 64 \text{ payload-bit features}\}$$

In contrast, our re-implementation adopts a *binary* classification setting, consistent with the teacher-guided adversarial training approach proposed in this thesis. The input feature vector is defined as $\{\texttt{ID}, \Delta t, \texttt{byte}_1, \ldots, \texttt{byte}_8\}$, where payload bytes are normalized and treated as continuous features. This formulation aligns with common practice in CAN IDS literature and enables the generation of semantically meaningful adversarial perturbations.

Under this binary formulation, the adversarial training objective follows the same rationale adopted in the teacher-guided strategy introduced in this thesis. In particular, training jointly optimizes classification performance on clean samples and on adversarially perturbed counterparts generated during training.

Formally, the resulting loss function can be expressed as:

$$\mathcal{L}_{self-adaptive} = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\frac{1}{1+|\mathcal{A}|} \left(\mathcal{L}(f(x), y) + \sum_{a\in\mathcal{A}} \mathcal{L}(f(x'_a), y) \right) \right],$$

where $\mathcal{A}$ is the set of adversarial attacks, x'_a denotes an adversarial example generated by applying attack $a \in \mathcal{A}$ directly to the model in its current training state, and $\mathcal{L}$ is the binary cross-entropy (BCE) loss.

Semantic consistency of adversarial examples A major correction concerns the semantic validity of adversarial samples. The original implementation relies on the external `pytorchattack` library, which assumes input features normalized to the $[0, 1]$ range and enforces the same constraint on perturbed outputs. However, the following issues arise:

- Input features are not explicitly normalized prior to invoking the attack library, resulting in `ID`, `DLC`, and inter-arrival time values that fall outside the assumed input domain.
- None of the employed features naturally conform to the enforced $[0, 1]$ output range:
  - payload-bit features are binary by definition and should remain in $\{0, 1\}$;
  - `ID` and `DLC` are integer-valued and span ranges far beyond $[0, 1]$;
  - inter-arrival time is artificially constrained to $[0, 1]$ without protocol-level justification.

Although these inconsistencies do not raise runtime errors, they lead to adversarial samples that violate protocol semantics and lack practical relevance.

To address these issues, we re-implement the attack algorithms locally, avoiding external attack libraries and enforcing realizability constraints directly during the perturbation process (see Section 4.2 for further details).

ID preprocessing An additional implementation error addressed in this work concerns the preprocessing of the CAN identifier. In the original pipeline, hexadecimal identifiers are converted to decimal values by removing alphabetic characters rather than performing a proper hexadecimal-to-decimal conversion. This procedure may induce collisions between distinct CAN IDs. Our implementation corrects this issue by applying a standard hexadecimal conversion, preserving identifier uniqueness.

Preservation of the dynamic ε schedule The dynamic scheduling of the attack budget ε, which progressively increases the maximum perturbation magnitude within each training epoch, is preserved from the original CANEDERLI methodology. This design choice is maintained to ensure methodological consistency, as the intra-epoch growth of ε constitutes a central component of the originally proposed adaptive adversarial training strategy.

Additional extensions In the original implementation, the adversarial training procedure relies on a fixed set of four attacks (FGSM, RFGSM, BIM, and PGD), which are jointly applied and cannot be selectively enabled, disabled, or replaced. Our implementation extends CANEDERLI adversarial training pipeline by introducing two additional hyperparameters also employed in the novel teacher-guided adversarial training approach: the set of adversarial attack methods used to generate perturbed samples, and the class balancing strategy (undersampling, oversampling, or class-weighted loss). These extensions are not intended to alter the core methodology of CANEDERLI adversarial training, but rather to provide greater flexibility and control to the practitioner when configuring the training process.

3.4. Model Architectures

To assess the impact of the proposed adversarial training strategies on baseline performance and robustness, this work considers three representative deep learning architectures commonly adopted in CAN-based intrusion detection systems: a Fully Connected Network (FCN), a Convolutional Neural Network (CNN), and a Long Short-Term Memory

(LSTM) network.

The choice of these architectures is motivated by their differing inductive biases and their ability to capture complementary aspects of CAN data, ranging from static message-level features to temporal dependencies across sequences of messages.

Each input sample is represented by a feature vector extracted from CAN traffic, consisting of:

- the eight payload bytes of the CAN message, normalized to the interval $[0, 1]$;
- the CAN identifier (`ID`);
- the Δt value, defined as the time elapsed since the previous message with the same `ID`.

This common input representation allows architectural differences to be isolated and analyzed independently of feature engineering choices.

Fully Connected Network (FCN). The FCN represents the simplest architectural baseline considered in this study. It processes each CAN message independently, without explicitly modeling temporal relationships between consecutive frames.

This architecture serves two key purposes. First, it provides a reference point for evaluating the effect of adversarial training in the absence of strong architectural priors. Second, its limited representational capacity makes it particularly useful for highlighting failure modes and sensitivity to adversarial perturbations, especially when applied to complex datasets. As such, the FCN enables an analysis of whether adversarial training alone is sufficient to compensate for architectural simplicity.

Convolutional Neural Network (CNN). The CNN architecture is designed to capture local temporal patterns in CAN traffic by operating on short windows of consecutive messages. By applying convolutional filters along the temporal dimension, the model learns to detect localized structures and correlations that may characterize attack behavior.

CNNs introduce an inductive bias toward local stationarity and pattern repetition, making them well suited for detecting short-lived or burst-like anomalies. Their inclusion in this study allows us to investigate how adversarial training interacts with architectures that partially encode temporal information while maintaining relatively shallow memory.

Long Short-Term Memory Network (LSTM). The LSTM architecture explicitly models temporal dependencies by processing fixed-length windows of consecutive CAN messages and maintaining an internal memory state across the sequence. This enables the network to capture both short-term dynamics and longer-range temporal dependencies within traffic windows.

LSTMs are particularly relevant for intrusion detection scenarios in which malicious behavior unfolds over multiple messages or exhibits temporal structure that cannot be captured by local context alone. From an adversarial perspective, sequence-based models also introduce different attack surfaces and robustness characteristics compared to stateless architectures, making them an essential component of a comprehensive evaluation.

Rationale for architectural diversity. The combination of FCN, CNN, and LSTM architectures enables a systematic analysis of adversarial training effects across models with increasing temporal awareness and representational complexity. This architectural diversity is crucial for studying not only robustness improvements within a single model class, but also the transferability of adversarial examples and defense strategies across fundamentally different learning paradigms.

By evaluating adversarial training on architectures ranging from stateless message-level models to sequence-based models capturing local and long-range temporal dependencies, this work aims to draw conclusions that are not tied to a specific network design, but are instead representative of broader classes of CAN-based intrusion detection systems.

3.5. Adversarial Methods

This work adopts a set of gradient-based adversarial attacks as core components of both the adversarial training strategies and the robustness evaluation protocol. Rather than serving solely as evaluation tools, adversarial methods are explicitly integrated into the training process to shape model decision boundaries and improve resistance to evasion attacks.

The selected attacks span a range of perturbation strategies, from lightweight single-step methods to stronger iterative procedures. This design enables the adversarial training pipelines to be configured with different attack combinations, allowing exposure to diverse adversarial signals when multiple methods are employed, while preserving the flexibility to train against a single, well-defined threat model.

The following adversarial methods are implemented:

- Fast Gradient Sign Method (FGSM) and its non-sign variant (FGM);
- Randomized Fast Gradient Sign Method (RFGSM);
- Basic Iterative Method (BIM);
- Projected Gradient Descent (PGD);
- a DeepFool-inspired iterative attack.

The specific subset of attacks used during adversarial training is treated as a configurable experimental parameter and can range from a single method to multiple complementary attacks. In contrast, robustness evaluation is conducted using stronger iterative attacks, namely BIM, PGD, and DeepFool, which provide a more reliable approximation of worst-case adversarial behavior.

Common constraints and assumptions. The adversarial sample generation scripts operate on the flat and windowed input representations described in Section 3.4, consisting of payload bytes normalized to $[0, 1]$, together with the `ID` and Δt features. The generation process is guided by domain-specific constraints that ensure semantic and protocol consistency. These key design choices are summarized below.

- **Consistent input normalization:** payload bytes are normalized to the interval $[0, 1]$ during training, validation, and testing. The same normalization scheme is strictly preserved during adversarial sample generation, ensuring consistency between clean and adversarial inputs.
- **Selective perturbation of malicious samples:** adversarial perturbations are generated exclusively from samples originally labeled as malicious. This choice reflects a realistic attacker objective, modifying malicious traffic to evade detection by inducing misclassification as benign, and avoids introducing unnecessary noise by perturbing naturally benign samples. Formally, perturbations are generated from the subset

  $$\mathcal{S}_{\text{mal}} = \{x \mid y(x) = \texttt{malicious}\}.$$

  Moreover, adversarial modifications are applied *only* to the payload-byte features. Given an input vector

  $$x = [\texttt{ID},\ \Delta t,\ b_1, \ldots, b_8], \qquad b_i \in [0, 1],$$

  the attack scripts modify exclusively the sub-vector $b = (b_1, \ldots, b_8)$, while `ID` and Δt are left unchanged. This design preserves the semantic integrity of identifier and

timing information and restricts perturbations to fields that an adversary would realistically manipulate on the CAN bus.

- **In-loop quantization and clipping:** at each perturbation step, byte values are constrained to remain interpretable as valid CAN payload bytes. After computing a perturbed normalized byte value b'_i, the following operation is applied:

$$b'_i \leftarrow \frac{\text{round}\left(\text{clip}(b'_i, 0, 1) \cdot 255\right)}{255}.$$

  Quantization and clipping are enforced *within* the attack loop, rather than as a post-processing step. Performing quantization during iterative attacks ensures that all intermediate adversarial examples remain semantically realizable and prevents the exploitation of continuous-valued artifacts that would be infeasible to implement in real CAN traffic.

Single-step and randomized attacks. FGSM and FGM represent lightweight, single-step adversarial attacks. Despite their simplicity, these attacks remain relevant in practice due to their low computational cost and their frequent use in adversarial training pipelines.

RFGSM extends FGSM by introducing a random initialization within the perturbation budget prior to applying the gradient-based update. This randomization partially mitigates gradient masking effects and produces more diverse adversarial samples, making RFGSM particularly suitable for adversarial training scenarios.

Iterative gradient-based attacks. BIM and PGD represent stronger, iterative variants of FGSM that refine adversarial perturbations through multiple gradient updates. BIM applies repeated FGSM steps with a fixed step size, while PGD additionally incorporates random restarts and projection onto the feasible perturbation set.

PGD is widely regarded as a first-order adversary that approximates the worst-case perturbation within a bounded norm ball, and is therefore commonly adopted as a reference attack for robustness evaluation. Including both BIM and PGD enables a fine-grained analysis of robustness degradation as attack strength increases.

DeepFool-inspired attack. In addition to norm-bounded gradient-based attacks, this work implements an iterative DeepFool-inspired evasion strategy. The original DeepFool algorithm is designed to compute minimal adversarial perturbations under the assumption of continuous and unconstrained input spaces and locally linear decision boundaries.

In the context of CAN-based intrusion detection, these assumptions do not strictly hold.

Input features are discrete, semantically constrained, and only a subset of them (payload bytes) can be meaningfully perturbed. As a result, we adapt the DeepFool procedure through quantization, feature masking, and clipping operations that enforce protocol-level constraints and preserve semantic validity.

Due to these adaptations, the resulting attack does not satisfy the theoretical optimality guarantees of the original DeepFool formulation. Nevertheless, it remains a valuable tool for probing model sensitivity to small, decision-boundary-oriented perturbations under realistic constraints.

4 | Implementation Details

4.1. Model Architectures

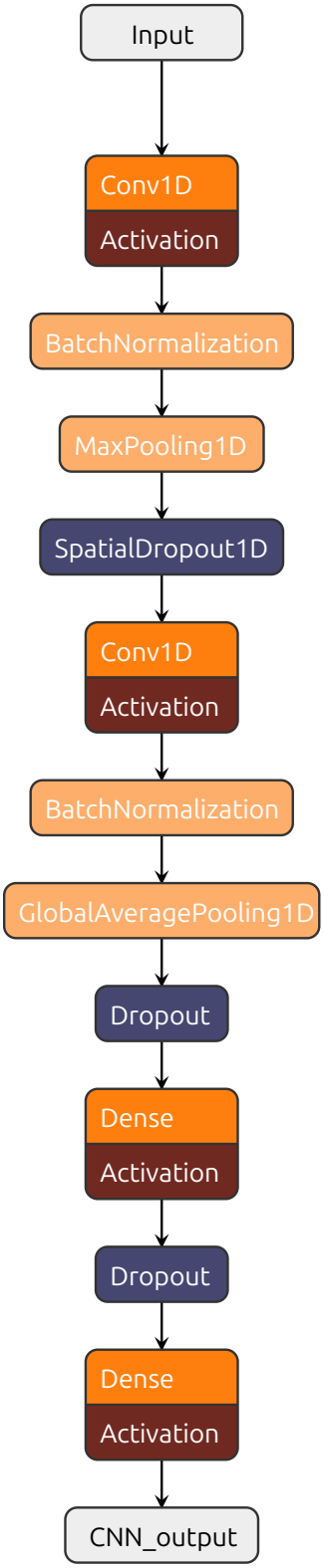
Three distinct neural architectures are implemented to assess the impact of the different training techniques considered in this work on baseline performance and adversarial robustness: a Fully Connected Network (FCN), a Convolutional Neural Network (CNN), and a Long Short-Term Memory (LSTM) network. All models are implemented in TensorFlow and trained as supervised binary classifiers to discriminate between *benign* and *malicious* CAN messages using labeled datasets.

From an implementation perspective, the architectures differ primarily in how temporal information is handled at the input level. The FCN operates on individual CAN messages and therefore receives single-frame feature vectors. In contrast, both the CNN and LSTM models are designed to exploit temporal context and are trained on fixed-length windows composed of 40 consecutive CAN messages.

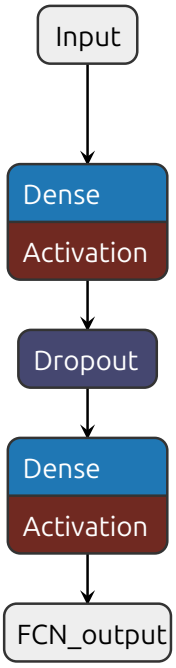
This window size represents a compromise between capturing sufficient temporal structure and maintaining computational compliance, and is kept consistent across all sequence-based models to ensure a fair comparison. Each window is treated as an independent training sample and labeled according to the presence of malicious activity within the corresponding message sequence. In particular, a window is labeled as malicious if at least one message in the sequence corresponds to an attack sample.

4.1.1. Fully Connected Network

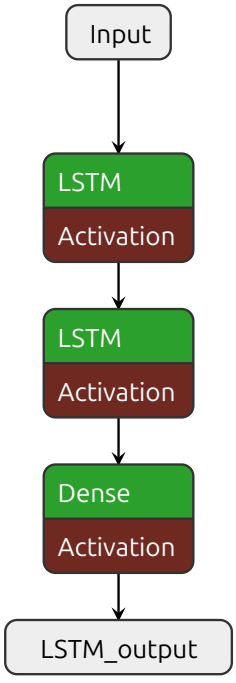
The network consists of a single fully connected hidden layer with ReLU activation, followed by a dropout layer with rate 0.3 applied during training to mitigate overfitting. The output layer employs a sigmoid activation function to produce class probabilities for binary classification.



(a) CNN architecture



(b) FCN architecture



(c) LSTM architecture

Figure 4.1: Comparison of the implemented network architectures. The diagrams illustrate the layer hierarchy and tensor flow for the CNN (left), FCN (top right), and LSTM (bottom right) models.

Formally, the FCN can be expressed as:

$$\hat{y} = \sigma\left(W_2 \,\text{Dropout}\left(\text{ReLU}(W_1 x + b_1)\right) + b_2\right),$$

where $x \in \mathbb{R}^d$ denotes the input feature vector of dimensionality d, and $\sigma(\cdot)$ is the sigmoid activation function.

This simple yet effective structure provides a useful benchmark for evaluating the impact of both architectural complexity and adversarial training strategies on robustness.

4.1.2. Convolutional Neural Network

The network receives as input a two-dimensional tensor of shape (T, F), where T denotes the window length and F the number of features per time step.

The architecture comprises two one-dimensional convolutional layers, each with 32 filters and a kernel size of 3, aimed at extracting local temporal patterns across consecutive CAN messages. Each convolution is followed by batch normalization to improve training stability and gradient flow. After the first convolutional block, a max-pooling layer is applied to reduce the temporal resolution, followed by a spatial dropout layer that randomly deactivates entire feature maps to enhance regularization.

After the second convolutional block, temporal information is aggregated using global average pooling, reducing the number of parameters compared to a flattening operation. The resulting representation is then processed by a fully connected hidden layer with 32 units and ReLU activation, followed by dropout. A final sigmoid-activated output neuron produces the binary classification score.

In addition to dropout-based regularization, L_2 weight regularization is applied to the convolutional kernels to further mitigate overfitting. This CNN architecture enables the model to capture short-term temporal dependencies in CAN traffic while maintaining a relatively compact parameterization.

4.1.3. Long Short-Term Memory Network

The network takes a $(T \times F)$ input tensor, where T and F correspond to the window size and the feature dimensionality, respectively.

The architecture consists of two stacked LSTM layers, each with 64 units and `tanh` activation. The first LSTM layer is configured to return the full output sequence (`return_sequences=True`), enabling the second layer to process the temporal evolution of the hidden

states across the entire window. The output of the second LSTM layer, corresponding to the final time step, is then fed into a dense layer with sigmoid activation to produce the binary classification score.

By preserving the temporal ordering of messages and maintaining an internal memory state, this architecture is well suited for time-dependent intrusion detection tasks.

4.2. Adversarial attack generation scripts

This section describes the implementation details of the adversarial attack generation pipeline introduced in Section 3.5. In particular, it focuses on the structure and behavior of the scripts used to generate adversarial CAN samples, the exposed configuration parameters, and the format of the produced outputs.

The attack generation scripts support multiple gradient-based adversarial methods, namely FGSM, RFGSM, BIM, PGD, and DeepFool. The specific attack methods are selected at runtime through a dedicated command-line parameter. In addition, users can specify the IDS model to act as source (or surrogate) model for adversarial sample generation. Further attack parameters exposed via the command-line interface include the perturbation budget ε, the number of attack iterations, and the step size, when applicable.

The adversarial attack generation pipeline is employed in two distinct modes, depending on the experimental setting.

During adversarial training, attacks are generated online and on-the-fly: for each training batch, adversarial samples are computed dynamically and immediately incorporated into the optimization process. In this setting, adversarial examples are not stored, as they are used exclusively to augment the current training batch.

Conversely, for adversarial testing and robustness evaluation, attacks are generated offline. In this case, adversarial samples are explicitly computed, validated for successful evasion of the source model, and then persisted to disk to form dedicated adversarial test datasets. More specifically, for each execution, the scripts produce an adversarial dataset containing only the malicious samples that are perturbed. The dataset is exported in a tabular format (i.e. CSV). Each row in the adversarial dataset corresponds to a single perturbed packet and contains the following fields:

- `original_index`: index of the packet in the original (clean) dataset;
- `id`: CAN identifier of the packet;
- Δt: inter-arrival time relative to the previous packet with the same `ID`;

- `byte_1`, ..., `byte_8`: the eight payload byte values after perturbation (represented in the normalized format used by the model, i.e., in $[0, 1]$);
- `perturbed_byte_1`, ..., `perturbed_byte_8`: absolute perturbation applied to each corresponding original byte value (in the same normalized scale);
- `perturbation`: aggregate perturbation magnitude for the packet, computed as the average of per-byte perturbations: $\frac{1}{8}\sum_{i=1}^{8}|\Delta b_i|$;
- `steps`: number of attack iterations required for this sample to evade the source model (value = 0 if the sample is already evasive; value = max_steps + 1 if no evasive sample can be generated within the imposed budget);
- `scenario`: label identifying the original CAN attack type associated with the malicious packet (e.g., DoS, fuzzing, RPM spoofing), inherited from the clean dataset.
- `oracle`: identifier (name) of the source/surrogate model used to generate the adversarial sample.

The resulting adversarial datasets are organized into separate CSV files according to the source model architecture, the attack method, and the selected perturbation budget.

4.2.1. Adversarial Attack Pseudocodes

To clarify the implemented attack procedures, we report:

- a compact pseudocode describing the main steps of the single-step FGM/FGSM routine (Algorithm 4.1);
- a compact pseudocode of the implemented deepfool procedure (Algorithm 4.2).

4.3. Adversarial Testing Scripts

Adversarial testing is performed by specifying both the adversarial attack method and the corresponding perturbation budget. The evaluation is conducted exclusively on adversarial samples that successfully evade the model used to generate them (i.e., the source or oracle model), ensuring that only effective adversarial examples are considered.

The testing pipeline is implemented through dedicated scripts that produce a summary table containing aggregated statistics and evaluation metrics. For each combination of oracle model, target model and attack scenario, the following fields are reported:

- `oracle`: identifier (name) of the source or surrogate model used to generate the

Algorithm 4.1 FGM/FGSM procedure for binary IDS models

Input: normalized sample x, model M, parameters $\{\varepsilon, \text{sign}, \text{idxs\_to\_perturb}, \text{max\_delta}\}$
Prepare input tensor x and original reference x_{orig}
Compute gradient of the loss with respect to x
Apply feature mask to restrict gradients to payload-byte features of malicious samples
if sign = True **then**
  $\delta = \varepsilon \cdot \text{sign}(\nabla_x L)$ # FGSM step
else
  $\delta = \varepsilon \cdot \frac{\nabla_x L}{\|\nabla_x L\|_2}$ # FGM step
end if
Apply perturbation and clip per-feature δ to $[-\text{max\_delta}, \text{max\_delta}]$
Quantize and clip perturbed byte features to the valid range $[0, 1]$
Return quantized adversarial sample x_{adv}

Note: Iterative attacks (BIM, PGD) are implemented as wrappers that repeatedly apply the single-step routine above, projecting the adversarial example onto the allowed perturbation budget after each iteration and enforcing quantization and clipping at every step.

adversarial samples;

- `target_model`: identifier (name) of the model on which the adversarial samples are evaluated;
- `scenario`: label identifying the original CAN attack type associated with the malicious packet (e.g., DoS, fuzzing, RPM spoofing), inherited from the clean dataset;
- `avg_steps_evasive_samples`: average number of optimization steps among the successful evasive adversarial samples;
- `avg_total_pert_evasive_samples`: average perturbation magnitude among the successful evasive adversarial samples;
- `max_pert`: maximum perturbation magnitude observed among the successful evasive adversarial samples;
- `TN, FP, FN, TP`: confusion matrix entries computed on the adversarial test set (True Negatives, False Positives, False Negatives, and True Positives, respectively). Note that `TN` and `FP` are always equal to zero, since the evaluation is conducted exclusively on malicious samples. Moreover, when the oracle and target models coincide, `TP` is always equal to zero, as the test set includes only adversarial samples that successfully evade the source model;

- `F1-score, accuracy, precision, recall`: standard classification performance metrics evaluated on adversarial samples.

Starting from these aggregated results, we derive the adversarial evaluation matrices and metrics described in Section 5.2.2. In particular, the testing scripts enable the computation of the following robustness and transferability indicators:

- white-box and grey-box robustness;
- mean perturbation magnitude required to evade a model under white-box and grey-box settings;
- mean number of optimization steps required to evade a model under white-box and grey-box settings;
- adversarial transferability across different model architectures.

Algorithm 4.2 DeepFool procedure for binary IDS models

Input: normalized sample x, model M, parameters $\{\varepsilon, \text{max\_iter}, \text{idxs\_to\_perturb}\}$

Initialize $x^{(0)} \leftarrow x$
Compute prediction $p = M(x^{(0)})$
if $p < 0.5$ **then**
  return x # already evasive
end if
for $t = 0$ to max_iter $- 1$ **do**
  Compute logit score $s = \log \frac{p}{1-p}$
  Compute gradient $g = \nabla_x s$
  Mask gradient to payload-byte features of malicious samples only
  Compute minimal perturbation:

$$\delta = -\frac{s}{\|g\|_2^2 + \epsilon} \cdot g$$

  Update adversarial sample:

$$x^{(t+1)} = x^{(t)} + (1 + \varepsilon)\,\delta$$

  Quantize perturbed byte features to valid discrete values
  Clip byte features to $[0, 1]$
  Compute new prediction $p = M(x^{(t+1)})$
  if $p < 0.5$ **then**
    return $x^{(t+1)}$ # successful evasion
  end if
end for
return x # attack failed

Note: The implementation follows the binary DeepFool formulation by linearizing the classifier decision function in logit space and enforcing feature-level realizability constraints at each iteration.

5 | Experimental Validation

5.1. Goals

The primary goal of this work is to investigate whether implementing adversarial training yields a meaningful improvement in robustness relative to the baseline detection performance. While defensive mechanisms, such as adversarial training, can increase the resilience of a model to crafted perturbations, they may also impact the model's accuracy on clean (non-adversarial) CAN traffic [3, 22]. In the context of automotive networks, where detection reliability on genuine traffic is critical, any improvement in adversarial robustness must be carefully weighed against potential degradations in baseline performance. This work systematically evaluates this trade-off by comparing models trained with and without adversarial training strategies, quantifying both their robustness under attack and their performance on standard traffic, to determine whether the gains justify the additional complexity of implementing such defenses.

To address this objective, the thesis revisits and extends the set of adversarial training strategies applicable to CAN-based intrusion detection systems. Specifically, it refines and re-implements the adaptive online adversarial training approach originally proposed in CANEDERLI [4] (hereafter referred to as *self-adaptive adversarial training*), and introduces a novel teacher-guided adversarial training strategy. These contributions aim to broaden the range of viable adversarial training options beyond conventional fine-tuning [1]. By expanding and systematizing the available training strategies, this work enables a more comprehensive evaluation of adversarial robustness in CAN-based IDSs and supports informed design choices when deploying defensive mechanisms in practice.

5.2. Extended Comparison of Training Strategies

5.2.1. Comparison Overview

To provide a comprehensive assessment of adversarial training in the CAN domain, three training paradigms are compared:

1. **Traditional Training:** The IDS model is trained exclusively on clean CAN data, without any form of adversarial augmentation. This serves as the baseline for both performance and robustness.
2. **Self-Adaptive adversarial training:** The re-implemented version of an existing CAN-specific adversarial training approach, incorporating the corrections and extensions described in the previous subsection.
3. **Teacher-Guided Adversarial Training:** The proposed method, where adversarial examples are generated by a separate teacher model and used to guide the training of the target IDS.

The comparison is conducted along two main evaluation dimensions:

- **Baseline Performance:** Accuracy and detection quality on unperturbed CAN traffic.
- **Adversarial Robustness:** Resistance to both white-box and grey-box attacks.

To enrich the analysis and gain deeper insight into how different models and training strategies interact in adversarial scenarios, adversarial transferability is also examined, i.e., the ability of adversarial examples generated for one model to deceive other models.

5.2.2. Comparison Details

This sub-section gives a precise definition of the metrics used to evaluate baseline performance, adversarial robustness, and adversarial transferability.

Baseline performance Baseline performance of each IDS model is measured on clean (unperturbed) test data using standard classification metrics derived from the confusion matrix. For a binary classification task we denote:

- TN = true negatives (benign correctly classified),
- FP = false positives (benign incorrectly flagged as malicious),
- FN = false negatives (malicious missed),
- TP = true positives (malicious correctly detected).

From these quantities we compute:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

In addition to these scalar metrics, the full confusion matrix is reported for every model to provide a complete view of classification errors.

Adversarial robustness and transferability matrices Adversarial evaluation is conducted considering three widely adopted attack algorithm: Basic Iterative Method (BIM), Projected Gradient Descent (PGD) and DeepFool. For each attack type k we construct a square matrix $S^{(k)}$ whose rows and columns index the set of evaluated models $\{M_1, M_2, \ldots, M_n\}$. The entry $S_{i,j}^{(k)}$ quantifies the *attack success rate* of adversarial examples generated using model M_j (as the source/surrogate) when tested against model M_i (as the target/victim), under attack type k. Formally, let D_j be the set of samples correctly classified as malicious by M_j during baseline testing. Let $N_j^{(k)}$ be the number of evasive adversarial samples generated from D_j using M_j as the source model and attack k. If $s_{i,j}^{(k)}$ of those adversarial samples cause model M_i to produce an incorrect (i.e., benign) decision, then the attack success rate is defined as:

$$S_{i,j}^{(k)} = \frac{s_{i,j}^{(k)}}{N_j^{(k)}} \in [0, 1].$$

Interpretation:

- Off-diagonal row values $S_{i,j\neq i}^{(k)}$ measure the **grey-box robustness** of model M_i against attacks crafted on other models trained on the same dataset, effectively representing the Grey-Box Attack Success Rate (ASR_{GB}). Rows with low values indicate models that are robust to attacks crafted on surrogate models.
- Off-diagonal column values $S_{i\neq j,j}^{(k)}$ capture the **adversarial transferability** of adversarial samples generated using model M_j: how effective M_j-generated attacks are against other models. Columns with generally high values indicate models that are strong attack generators.
- The diagonal entries $S_{i,i}^{(k)}$ represent the proportion of successful attacks generated from M_j that evade M_i, with M_j and M_i denoting the same model, therefore they are trivially equal to 1.

Finally, the **white-box robustness** of model M_j for attack k is measured using the White-Box Attack Success Rate (ASR_{WB}), which quantifies the fraction of initially cor-

rectly classified malicious samples that successfully evade the detection after the perturbation. Formally:

$$\mathrm{ASR}_{\mathrm{WB}}(M_j, k) \;=\; \frac{N_j^{(k)}}{|D_j|} \in [0,1].$$

Low values of $\mathrm{ASR}_{\mathrm{WB}}$ indicate a higher robustness of model M_j against the self-crafted attack k, as they represent a lower proportion of evasive perturbed samples obtained from M_j.

In addition to rate-based matrices, for each successful adversarial sample we record several per-sample quantities. Aggregating these per-sample values per *source* model M_j provides additional insight into the *effort* required to bypass a given model in a white-box scenario:

- **Mean number of optimization steps to success for model M_j under attack k**:

$$\overline{Steps}_j^{(k)} \;=\; \frac{1}{s_{j,j}^{(k)}} \sum_{t=1}^{s_{j,j}^{(k)}} steps_{j,j,t}^{(k)},$$

  where $steps_{j,j,t}^{(k)}$ is the number of attack iterations required by the t-th successful adversarial sample generated from M_j.

- **Mean perturbation magnitude for successful samples that evade M_j**:

$$\overline{\Delta}_j^{(k)} \;=\; \frac{1}{s_{j,j}^{(k)}} \sum_{t=1}^{s_{j,j}^{(k)}} \|\delta_{j,t}^{(k)}\|_\infty,$$

  where $\delta_{j,t}^{(k)}$ is the perturbation applied to obtain the t-th successful adversarial sample and $\|\cdot\|_\infty$ denotes the L_∞ norm.

We also compute symmetric statistics from the perspective of each *target* model M_i. These metrics summarize the characteristics of successful adversarial samples *that actually evades* M_i and are generated by surrogate models M_j with $j \neq i$, thus enabling a more comprehensive evaluation of robustness under grey-box conditions.

Let

$$S_i^{(k),succ} \;=\; \sum_{j \neq i} s_{i,j}^{(k)}$$

be the total number of successful adversarial samples that bypass model M_i under attack k, where $s_{i,j}^{(k)}$ is the number of successful samples generated from surrogate M_j that fools M_i.

We then define the following target-centric aggregates:

- **Mean number of optimization steps to success for target M_i:**

$$\overline{Steps}_{i,target}^{(k)} = \frac{1}{S_i^{(k),succ}} \sum_{j \neq i} \sum_{t=1}^{s_{i,j}^{(k)}} steps_{i,j,t}^{(k)},$$

  where $steps_{i,j,t}^{(k)}$ is the number of attack iterations required by the t-th successful sample generated from M_j that evades M_i.

- **Mean perturbation magnitude for successful samples that evade M_i:**

$$\overline{\Delta}_{i,target}^{(k)} = \frac{1}{S_i^{(k),succ}} \sum_{j \neq i} \sum_{t=1}^{s_{i,j}^{(k)}} \|\delta_{i,j,t}^{(k)}\|_\infty,$$

  where $\delta_{i,j,t}^{(k)}$ is the perturbation applied to the t-th successful sample (generated from M_j) that evades M_i.

Practical considerations

- **Success criterion.** In our binary IDS setting an adversarial sample is considered successful if it causes a malicious input to be classified as benign (i.e., a false negative). All rates $S_{i,j}^{(k)}$ are computed with this criterion.
- **Attack budgets and constraints.** For each attack algorithm k we fix perturbation budgets and iteration parameters consistent with feasible modifications in the CAN domain; these experimental settings are reported in Section 5.6.

5.3. Description of the Datasets

In this study, we utilize two datasets: the well-known `HCRL Survival Analysis` dataset, curated by Mee Lan Han, Byung Il Kwak, and Huy Kang Kim [23], and the newly introduced `can-train-and-test` dataset, curated by Brooke Lampe and Weizhi Meng [24–26].

5.3.1. HCRL Survival Analysis Dataset

The `HCRL Survival Analysis` dataset was published by the Korea University (HCRL) as part of research on vehicular intrusion detection and survival analysis-based detection methods [23]. It was designed to provide a real-world benchmark for evaluating IDSs

using dynamic driving data and multiple attack scenarios across different vehicles.

Data Sources and Structure The dataset comprises CAN traffic collected from three distinct vehicles: a Chevrolet Spark, a Hyundai YF Sonata, and a Kia Soul. It includes both attack-free and attack-injected traffic captures recorded during normal driving conditions. Each recording contains a number of standard CAN features, typically including message timestamps, arbitration IDs, data length codes (DLC), payload bytes, and binary labels indicating whether a message is malicious or benign.

The dataset is structured to support both training and evaluation of IDS models on data from multiple vehicles and multiple attack types. For the purposes of comparative evaluation, the dataset can be partitioned into training and testing splits based on vehicle identity and attack scenarios, facilitating assessments of generalization across vehicles and attack conditions.

Key Statistics and Attack Scenarios The HCRL Survival Analysis dataset includes both benign traffic and three primary attack types:

- **Flooding (Denial-of-Service):** high-frequency injection of messages with a dominant arbitration ID to saturate the bus and prevent normal communication.
- **Fuzzing:** injection of malformed or randomly generated CAN frames intended to elicit anomalous responses from ECUs.
- **Malfunction (Spoofing):** crafted messages that simulate erroneous sensor values or abnormal operational states.

Unlike some other open CAN IDS datasets, this dataset contains only real, on-road driving data for each vehicle under both normal operation and attack conditions, increasing its relevance for realistic evaluation.

Availability and Access The HCRL Survival Analysis dataset is publicly accessible through the HCRL dataset repository hosted on the official HCRL website, which provides download links for academic use: `https://ocslab.hksecurity.net/Datasets/survival-ids`. Researchers can obtain the dataset along with documentation detailing its attributes, recording methodology, and attack scenarios.

Strengths of the Dataset The HCRL Survival Analysis dataset offers several advantages:

- **Multi-vehicle data:** traffic captured from three different vehicles supports evaluations of cross-vehicle generalization.
- **Real driving conditions:** attacks and benign operational behavior are recorded during actual vehicle motion rather than stationary conditions.
- **Message-level labeling:** each CAN message is annotated as benign or malicious, supporting supervised and semi-supervised learning paradigms.

Key Limitations Despite its utility, several limitations of the dataset have been identified:

- **Limited data volume:** the total number of samples per vehicle and per attack class is relatively small compared to more recent datasets, which can hinder the training of data-intensive machine learning models.
- **Basic attack implementations:** attack behaviors are relatively straightforward and may not capture the complexity of attacks deployed by modern adversaries.

Overall, the HCRL Survival Analysis dataset remains a widely cited reference in the CAN intrusion detection literature and provides an important complement to more recent, larger-scale datasets, even though its limited volume and simpler attack profiles necessitate careful interpretation of results.

5.3.2. CAN-train-and-test Dataset

The `can-train-and-test` dataset [24–26] was introduced to support the development and evaluation of machine learning–based Intrusion Detection Systems (IDSs) for the Controller Area Network (CAN) protocol. It was designed to address the longstanding lack of publicly available, realistic CAN datasets, which has traditionally limited reproducibility and comparative analysis in automotive cybersecurity research.

Data Sources and Structure The `can-train-and-test` dataset consists of CAN traffic collected from four distinct vehicles produced by two different manufacturers: General Motors (Chevrolet) and Subaru. The selected vehicles span multiple categories, including a sedan, a compact SUV, a full-size SUV, and a pickup truck. For each vehicle model, the dataset provides comparable benign and attack traffic captures, enabling the evaluation of IDS generalization across different vehicle types and manufacturers.

To accommodate diverse research needs, the dataset is released in multiple formats:

- **Replayable `.log` files**: raw CAN bus traffic suitable for exact traffic replay and low-level analysis.
- **Labeled `.csv` files**: CAN messages annotated as benign or malicious, supporting supervised learning approaches.
- **Unlabeled `.csv` files**: CAN messages without annotations, intended for unsupervised or semi-supervised methods.

Key Statistics and Attack Scenarios The dataset includes nine distinct attack types, ranging from Denial-of-Service (DoS) attacks to various spoofing attacks, such as gear spoofing and standstill injection. Notably, several of these attacks, particularly the spoofing-based ones, were executed during live, on-road experiments using real vehicles, ensuring that the injected traffic produces observable physical effects. This characteristic significantly enhances the realism and practical relevance of the dataset.

Each vehicle-specific subset is further partitioned into training and testing splits, enabling controlled evaluation scenarios. This organization allows models to be trained on a subset of vehicles or attack types and tested on previously unseen configurations, thereby facilitating the assessment of cross-vehicle and cross-attack generalization capabilities.

Availability and Access The `can-train-and-test` dataset is publicly available and can be accessed through the following repositories:

- `https://data.dtu.dk/articles/dataset/can-train-and-test/24805533`
- `https://bitbucket.org/brooke-lampe/can-train-and-test-v1.5`

Strengths of the Dataset The `can-train-and-test` dataset is presented as one of the most comprehensive and high-fidelity open-source datasets currently available for automotive intrusion detection research. While it offers several significant advantages, it also exhibits limitations that should be carefully considered when designing and evaluating CAN-based intrusion detection systems.

The dataset exhibits several notable strengths:

- **Vehicle diversity:** CAN traffic is collected from four different vehicles produced by two manufacturers, enabling the evaluation of IDS models across heterogeneous automotive platforms.
- **Attack variety:** multiple attack types are included, spanning denial-of-service, spoofing, and standstill attacks.

- **Extensive benign traffic:** large portions of the dataset consist of attack-free CAN traffic, providing a reliable baseline for normal vehicle behavior.
- **High-fidelity data collection:** several attacks were executed on real vehicles rather than in simulated environments, resulting in realistic traffic patterns with known physical effects.
- **Message-level labeling:** each CAN message is annotated as benign or malicious, supporting supervised and semi-supervised learning paradigms.

Key Limitations A primary limitation of the dataset is the low variability of malicious payloads within individual attack traces. In most attack-specific files, a single malicious payload is repeated multiple times, with the inter-arrival time (Δt) representing the only varying feature between consecutive malicious messages. Across the entire dataset, for a given vehicle and attack type, only four distinct malicious payloads are present: two in the training subset and two in the testing subset (specifically within the `test_01_known_vehicle_known_attack` split).

This constrained diversity introduces several challenges:

- **Limited supervised generalization:** models trained on such data must infer an attack pattern from only two unique payload instances, which hinders the learning of robust representations capable of generalizing to unseen payload variations.
- **Evaluation bias:** evaluation against a given attack effectively exposes the model to only two distinct malicious payloads. Consequently, the model typically either detects both payloads, fails on both, or detects only one of the two, yielding performance results that are not necessarily representative of its true detection capability.

An additional limitation concerns the temporal concentration of malicious traffic within individual logs. In most recordings, attack packets are confined to short, contiguous segments, while the remainder of the trace consists of benign traffic. This uneven distribution complicates the creation of training and validation splits, as naive partitioning strategies may introduce data leakage or produce highly imbalanced subsets. Such constraints reduce partitioning flexibility and may negatively impact training stability and evaluation consistency.

Overall, while the `can-train-and-test` dataset provides a valuable and realistic benchmark for CAN IDS research, both the limited intra-attack payload diversity and the localized temporal concentration of malicious packets must be taken into account when interpreting experimental results, particularly with respect to baseline performance and

generalization capability.

5.4. Experimental Setup

5.4.1. Threat Model

The threat model considered in this work reflects a realistic in-vehicle deployment scenario for an Intrusion Detection System (IDS). Specifically, the IDS is assumed to operate on-board a vehicle, monitoring only the CAN traffic generated by that vehicle. This assumption is motivated by the nature of the CAN protocol: message identifiers (`IDs`) and payload encodings differ significantly across vehicle models, even within the same manufacturer. Consequently, a message or attack valid for one vehicle type is unlikely to be syntactically or semantically valid on another.

Under this premise, both training and evaluation of each IDS model are restricted to CAN traffic corresponding to the target vehicle model. Cross-vehicle generalization (i.e., detecting attacks on unseen vehicle types) is therefore considered out of scope, as it does not correspond to a realistic deployment scenario. Accordingly, the evaluation subsets `test_02_unknown_vehicle_known_attack` and `test_04_unknown_vehicle_unknown_attack` from the *can-train-and-test* dataset are excluded from all experiments.

Given the limited data volume of the `HCRL Survival Analysis` dataset and the limited diversity of malicious payloads in the `can-train-and-test` dataset (see Section 5.3), all robustness and transferability evaluations are performed under **white-box** and **grey-box** settings:

- In the *white-box* setting, the adversary has full knowledge of the target model, including its parameters and training data distribution.
- In the *grey-box* setting, the adversary has access only to the dataset used to train the target model, without knowledge of the model parameters.

A black-box setting is deliberately avoided. For one dataset, it would further reduce the effective training set size; for the other, it would further limit the diversity of malicious payloads available during training, potentially yielding less reliable results. This design choice is consistent with prior work. Notably, Longari et al., [18] reports no substantial differences in adversarial robustness outcomes when assuming dataset-level knowledge (grey-box) versus zero knowledge (black-box).

5.4.2. Experimental Insights

Dataset split and scope The experimental evaluation conducted in this thesis considers all available vehicles from the `HCRL Survival Analysis` dataset and two vehicles from the `can-train-and-test` dataset, namely the Chevrolet Impala and the Subaru Forester.

Since the two datasets exhibit fundamentally different structures and data organization, they are handled using distinct splitting strategies. For the `HCRL Survival Analysis` dataset, each vehicle is associated with four CAN trace files. For each file, a temporal split is applied as follows: the first 70% of the trace is used for training, the subsequent 10% for validation, and the remaining 20% for testing. This approach preserves the temporal ordering of CAN traffic and avoids information leakage across splits. For the `can-train-and-test` dataset, training and evaluation follow the official dataset partitions provided by the authors, in accordance with the threat model defined in Section 5.4.1. Specifically, for each selected vehicle:

- **Training set:** the official training split, denoted as `train_01`;
- **Validation set:** a fixed held-out subset corresponding to 20% of the original training split;
- **Test set:** the `test_01_known_vehicle_known_attack` subset, which contains previously unseen attack instances performed on the same vehicle.

Training procedure The training procedure is kept consistent across all experimental runs, with variations only arising from the chosen model architecture and training strategy (standard training, teacher-guided adversarial training, or adaptive online adversarial training, as described in Chapter 3). The main aspects of the training setup are summarized below:

- All models are optimized using the Adam optimizer with a fixed learning rate of 10^{-4}.
- The binary cross-entropy loss is employed for all experiments, reflecting the binary nature of the intrusion-detection task.
- A batch size of 128 is used across all training runs.
- Models are trained until convergence using an early-stopping policy monitored on the validation set; the patience is set to 8 epochs, with a maximum of 30 training epochs.

- For each model and configuration, the checkpoint corresponding to the best validation performance is retained for subsequent evaluation.
- For sequence-based architectures (CNN and LSTM), the temporal window length is fixed to 40 consecutive CAN messages.

For adversarially trained models, the maximum perturbation budget used during training is fixed to $\varepsilon = 0.2$. This choice ensures that the models are exposed to sufficiently strong adversarial perturbations during training, while remaining within the bounds of semantically realizable CAN payload modifications. Moreover, the adversarial attacks employed during training include BIM, FGSM, PGD, and RFGSM, in accordance with the original CANEDERLI implementation [4]. For the iterative attacks (BIM and PGD), the maximum number of iterations is fixed to $n_{\max} = 10$ and the per-step perturbation magnitude is defined as

$$\alpha = \frac{\varepsilon}{n_{\max}},$$

5.5. Experiment 1: Baseline Performance

5.5.1. Description of the Experiment

The first research question addressed in this experimental evaluation concerns the potential drawbacks of adopting adversarial training strategies in CAN-based intrusion detection systems. While adversarial training is widely recognized as an effective defense against evasion attacks, a well-documented side effect is the degradation of baseline performance on clean, non-adversarial data [3, 22].

In safety-critical domains such as automotive networks, maintaining high detection accuracy on legitimate CAN traffic is essential. A reduction in baseline performance may result in increased false positives or missed detections, undermining the practical deployability of an IDS, even if adversarial robustness is improved. For this reason, any robustness gain introduced by adversarial training must be carefully evaluated against its impact on standard detection performance.

To quantify this trade-off, we evaluate and compare models trained with and without adversarial training exclusively on unperturbed test data. All models are assessed following the experimental setup described in Section 5.4, ensuring a fair and consistent comparison across architectures, datasets, and training strategies. This experiment establishes a reference point for subsequent robustness and transferability analyses, allowing adversarial performance to be interpreted in light of the corresponding baseline behavior.

Baseline performance is quantified using standard classification metrics derived from the confusion matrix. Specifically, for each model and dataset, we report the confusion matrix entries (TN, FP, FN, TP), together with the corresponding accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of the detection behavior on clean CAN traffic, capturing both overall classification performance and the balance between false positives and false negatives, where higher values indicate better performance for all metrics.

5.5.2. Discussion of the observed outcomes and their implications

Contrary to the common expectation that adversarial training necessarily degrades baseline detection performance, the results obtained in this experiment indicate that, in most cases, models trained with adversarial strategies achieve baseline performance that is comparable to, or even slightly better than, that of their standardly trained counterparts.

When analyzing individual vehicles and architectures, the confusion matrices and derived metrics show no systematic degradation in accuracy, precision, recall, or F1-score for adversarially trained models. In several configurations, both the teacher-guided and the adaptive online adversarial training strategies yield baseline performance that closely matches that of standard training, with only marginal fluctuations that fall within normal training variability.

A closer inspection of the per-vehicle results reveals, however, several cases of particular interest that help contextualize the aggregate trends. In particular, the two vehicles drawn from the `can-train-and-test` dataset (Chevrolet Impala and Subaru Forester) exhibit a noticeably higher level of complexity compared to the vehicles included in the `HCRL Survival Analysis` dataset. This increased complexity is reflected in consistently lower baseline performance for simpler model architectures.

For both the Impala (Table 5.4) and the Forester (Table 5.5), Fully Connected Networks (FCNs) appear insufficient to correctly model the underlying data distribution, regardless of the training strategy employed. The confusion matrices indicate that these models struggle to define meaningful decision boundaries, resulting in poor detection capability. In the case of the Impala dataset, this limitation is particularly evident: both the standard FCN and the teacher-guided FCN fail to identify any malicious samples, yielding zero true positives (TP = 0).

This observation highlights an important aspect of the teacher-guided adversarial training

strategy that must be taken into account when interpreting its performance. In the proposed framework, teacher-guided training relies on a surrogate *teacher* model that is itself trained using standard training. Consequently, the effectiveness of the student model is inherently bounded by the generalization capability of the corresponding baseline model. When the teacher fails to learn a meaningful decision boundary, as observed for FCNs on more complex datasets, the adversarial signals provided during teacher-guided training are of limited utility and may even become detrimental, as the resulting student model inherits and potentially amplifies the same weaknesses.

These results emphasize that teacher-guided adversarial training should not be viewed as a mechanism for compensating architectural inadequacy. Rather, it is most effective when applied to models that already exhibit a reasonable baseline performance, where adversarial guidance can meaningfully refine the learned decision boundaries. This dependency further motivates the use of sufficiently expressive architectures when deploying teacher-guided adversarial training in complex CAN environments.

model	**method**	**TN**	**FP**	**FN**	**TP**	**accuracy**	**precision**	**recall**	**f1-score**
FCN	standard	89861	1369	690	15090	0.9808	0.9168	0.9563	0.9361
	self-adaptive	90939	291	225	15555	0.9952	0.9816	0.9857	0.9837
	teacher-guided	91016	214	286	15494	0.9953	0.9864	0.9819	0.9841
CNN	standard	1575	0	0	1098	1	1	1	1
	self-adaptive	1574	1	0	1098	0.9996	0.9991	1	0.9995
	teacher-guided	1575	0	0	1098	1	1	1	1
LSTM	standard	1575	0	0	1098	1	1	1	1
	self-adaptive	1575	0	0	1098	1	1	1	1
	teacher-guided	1575	0	0	1098	1	1	1	1

Table 5.1: Baseline detection performance on clean CAN traffic – Hyundai Sonata

model	method	TN	FP	FN	TP	accuracy	precision	recall	f1-score
FCN	standard	138537	789	9246	10999	0.9371	0.9331	0.5433	0.6867
	self-adaptive	139110	216	684	19561	0.9944	0.9891	0.9662	0.9775
	teacher-guided	138985	341	667	19578	0.9937	0.9829	0.9671	0.9749
CNN	standard	2400	0	0	1587	1	1	1	1
	self-adaptive	2399	1	0	1587	0.9997	0.9994	1	0.9997
	teacher-guided	2398	2	0	1587	0.9995	0.9987	1	0.9994
LSTM	standard	2400	0	2	1585	0.9995	1	0.9987	0.9994
	self-adaptive	2400	0	2	1585	0.9995	1	0.9987	0.9994
	teacher-guided	2400	0	2	1585	0.9995	1	0.9987	0.9994

Table 5.2: Baseline detection performance on clean CAN traffic – Kia Soul

model	method	TN	FP	FN	TP	accuracy	precision	recall	f1-score
FCN	standard	73195	2	1026	6369	0.9872	0.9997	0.8613	0.9253
	self-adaptive	71629	1568	0	7395	0.9805	0.8251	1.0000	0.9041
	teacher-guided	71547	1650	0	7395	0.9795	0.8176	1.0000	0.8996
CNN	standard	1501	2	1	508	0.9985	0.9961	0.9980	0.9971
	self-adaptive	1473	30	1	508	0.9846	0.9442	0.9980	0.9704
	teacher-guided	1495	8	1	508	0.9955	0.9845	0.9980	0.9912
LSTM	standard	1498	5	1	508	0.9970	0.9903	0.9980	0.9941
	self-adaptive	1498	5	1	508	0.9970	0.9903	0.9980	0.9941
	teacher-guided	1484	19	1	508	0.9901	0.9639	0.9980	0.9807

Table 5.3: Baseline detection performance on clean CAN traffic – Chevrolet Spark

model	method	TN	FP	FN	TP	accuracy	precision	recall	f1-score
FCN	standard	5639390	0	63280	0	0.9889	0.0000	0.0000	0.0000
	self-adaptive	5618018	21372	42557	20723	0.9888	0.4923	0.3275	0.3933
	teacher-guided	5639390	0	63280	0	0.9889	0.0000	0.0000	0.0000
CNN	standard	137818	2	1448	3294	0.9898	0.9994	0.6946	0.8196
	self-adaptive	137816	4	1265	3477	0.9911	0.9989	0.7332	0.8457
	teacher-guided	137802	18	1202	3540	0.9914	0.9949	0.7465	0.8530
LSTM	standard	127473	10347	1021	3721	0.9203	0.2645	0.7847	0.3956
	self-adaptive	137691	129	925	3817	0.9926	0.9673	0.8049	0.8787
	teacher-guided	137716	104	1077	3665	0.9917	0.9724	0.7729	0.8612

Table 5.4: Baseline detection performance on clean CAN traffic – Chevrolet Impala

model	method	TN	FP	FN	TP	accuracy	precision	recall	f1-score
FCN	standard	5371200	609401	48878	8660	0.8910	0.0140	0.1505	0.0256
	self-adaptive	5407899	572702	49461	8077	0.8970	0.0139	0.1404	0.0253
	teacher-guided	5842297	138304	41879	15659	0.9702	0.1017	0.2722	0.1481
CNN	standard	131802	131	16319	2696	0.8910	0.9537	0.1418	0.2469
	self-adaptive	131774	159	15408	3607	0.8969	0.9578	0.1897	0.3167
	teacher-guided	130676	1257	14553	4462	0.8953	0.7802	0.2347	0.3608
LSTM	standard	130110	1823	12456	6559	0.9054	0.7825	0.3449	0.4788
	self-adaptive	131898	35	13482	5533	0.9105	0.9937	0.2910	0.4501
	teacher-guided	131332	601	13410	5605	0.9072	0.9032	0.2948	0.4445

Table 5.5: Baseline detection performance on clean CAN traffic – Subaru Forester

model	method	accuracy	f1-score
FCN	standard	0.9570	0.5147
	self-adaptive	0.9712	0.6568
	teacher-guided	0.9855	0.6013
CNN	standard	0.9759	0.8127
	self-adaptive	0.9744	0.8264
	teacher-guided	0.9763	0.8409
LSTM	standard	0.9644	0.7736
	self-adaptive	0.9799	0.8645
	teacher-guided	0.9777	0.8572

Table 5.6: Mean baseline detection performance on clean CAN traffic

More notably, when considering the aggregated results across all evaluated vehicles and models (Table 5.6), adversarially trained models exhibit a consistent improvement in baseline detection performance. This trend suggests that, in the considered setting, adversarial training does not merely preserve clean-data performance but can also act as an implicit regularization mechanism.

A plausible explanation for this behavior is that the introduction of small, constrained adversarial perturbations during training encourages the models to learn smoother decision boundaries and more robust feature representations. Even though the perturbations are designed to challenge the classifier, their limited magnitude ensures that the semantic validity of CAN messages is preserved. As a result, the models may generalize better to previously unseen but legitimate variations in clean traffic, reducing overfitting to the specific patterns present in the training data.

> *These findings have important practical implications. They suggest that, at least for the perturbation budgets and training strategies considered in this work, adversarial training does not impose the expected cost in terms of baseline performance. Instead, it can be integrated into CAN-based IDS pipelines without compromising, and in some cases even improving, detection reliability on non-adversarial traffic.*

This observation strengthens the motivation for adopting adversarial training in automotive IDSs and provides a solid baseline for interpreting the robustness and transferability

results presented in the subsequent experiment.

5.6. Experiment 2: Adversarial Robustness and Transferability

5.6.1. Description of the Experiment

The second research question addressed in this experimental validation concerns the extent to which adversarial training strategies improve the robustness of CAN-based intrusion detection systems against adversarial manipulation. While the previous experiment focused on quantifying the impact of adversarial training on baseline detection performance, this experiment aims to assess whether the proposed training strategies effectively enhance resistance to evasion attacks and improve adversarial robustness under realistic threat models.

In particular, this experiment evaluates both *white-box* (WB) and *grey-box* (GB) robustness, as well as the transferability of adversarial examples across different model architectures and training strategies. Robustness improvements are quantified by measuring the ability of adversarially trained models to withstand carefully crafted perturbations that successfully evade a source (oracle) model.

Adversarial robustness and transferability are assessed using adversarial examples generated by three attack methods implemented in the attack-generation scripts described in Section 4.2: the Basic Iterative Method (BIM), Projected Gradient Descent (PGD), and DeepFool.

The evaluation considers multiple attack–budget configurations in order to capture the effect of different perturbation strengths. Specifically, adversarial samples are generated using:

- BIM with $\varepsilon = 0.1$,
- BIM with $\varepsilon = 0.2$,
- PGD with $\varepsilon = 0.1$,
- PGD with $\varepsilon = 0.2$,
- DeepFool with $\varepsilon = 0.01$.

The maximum number of attack iterations is fixed to $n_{\max} = 50$ for all methods.

For the iterative attacks (BIM and PGD), the per-step perturbation magnitude is defined as

$$\alpha = \frac{\varepsilon}{n_{\max}},$$

ensuring that the cumulative perturbation remains bounded by the prescribed budget.

For DeepFool, the reported ε does not represent a strict perturbation bound, but corresponds to the *overshoot* parameter, which controls the amount by which the final adversarial perturbation exceeds the estimated decision boundary. The attack terminates either upon successful evasion or upon reaching the maximum number of iterations.

All adversarial evaluations are conducted exclusively on adversarial samples that successfully evade the source (oracle) model, in accordance with the threat model and evaluation methodology described in Section 5.4.1. This choice ensures that robustness and transferability are measured only on effective adversarial examples, avoiding confounding effects introduced by unsuccessful attacks.

Furthermore, adversarial perturbations are generated only from malicious samples that are correctly detected by the source model in the baseline (clean) evaluation. That is, only samples that are initially classified as malicious are selected for adversarial manipulation and subsequent testing. As a result, any successful evasion observed during adversarial evaluation can be unambiguously attributed to the introduced perturbation, ensuring that the generated adversarial examples represent meaningful and high-quality attacks rather than artifacts of pre-existing model errors.

As described in Section 5.2.2, for each implemented attack we compute three evaluation matrices:

- the *Attack Success Rate (ASR)* matrix, reporting the percentage of evasive adversarial samples generated using the source model identified by the column that successfully evade the target model identified by the row;
- the *perturbation* matrix, reporting the mean perturbation magnitude (expressed as a percentage) required by evasive adversarial samples generated by the model of the column to evade the model of the row (e.g., a value of 0.1 indicates an average perturbation of 10% with respect to the original sample);
- the *steps* matrix, reporting the mean number of optimization steps required by evasive adversarial samples generated by the model of the column to evade the model of the row.

For each vehicle, this results in a total of 15 matrices (three metrics for each of the

five evaluated attacks). These matrices are subsequently aggregated and synthesized per attack and per vehicle to obtain multiple levels of granularity, enabling analyses at different abstraction levels and facilitating a clear and immediate interpretation of the experimental results.

It is worth noting that an empty column in an ASR matrix indicates that no successful evasive adversarial samples were generated using the corresponding source model. This situation can arise under two distinct conditions.

First, the source model may fail to correctly detect any malicious samples during baseline evaluation, resulting in no eligible inputs for adversarial perturbation. Second, although the source model correctly detects malicious samples, the applied adversarial attack is not able to generate perturbations that successfully evade the source model within the imposed constraints.

These two cases can be unambiguously distinguished by considering the baseline performance of the source model. In particular, when a model exhibits zero detection capability (e.g., zero accuracy and zero F1-score on malicious samples), the former condition applies, otherwise, an empty column reflects the inability of the considered adversarial methods to induce successful evasion.

5.6.2. Discussion of the observed results

We begin by discussing the aggregated robustness results obtained across all vehicles and attack methods, as summarized in Table 5.7. This high-level view allows us to identify global trends across architectures and training strategies before analyzing finer-grained behaviors.

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	0.1148	0.0898	16.66	0.8686	0.0919	18.01	0.2274
	self-adaptive	0.2679 (+133%)	0.0777	16.39	0.5089 (-41%)	0.1104	19.76	0.3737 (+64%)
	teacher-guided	0.2296 (+100%)	0.0703	14.30	0.5121 (-41%)	0.1010	19.03	0.2600 (+14%)
CNN	standard	0.1057	0.0899	19.00	0.4119	0.1002	18.18	0.2953
	self-adaptive	0.0460 (-56%)	0.1124	13.59	0.2638 (-36%)	0.1163	13.45	0.5609 (+90%)
	teacher-guided	0.0535 (-49%)	0.0845	23.66	0.2444 (-41%)	0.1107	14.71	0.3741 (+27%)
LSTM	standard	0.0814	0.1093	17.86	0.2336	0.0976	17.16	0.4446
	self-adaptive	0.0462 (-43%)	0.1096	18.91	0.1737 (-26%)	0.1237	12.70	0.5557 (+25%)
	teacher-guided	0.0448 (-45%)	0.0998	18.03	0.2084 (-11%)	0.0931	18.11	0.4275 (-3.8%)

Table 5.7: Mean adversarial performance on perturbed CAN traffic - global

Table 5.7 reveals a clear and consistent trend: the adoption of adversarial training substan-

tially improves robustness against adversarial attacks across all considered architectures. In particular, the grey-box (GB) evaluation highlights significant gains for models whose baseline adversarial performance was initially low. Both FCN- and CNN-based architectures exhibit an overall increase in grey-box robustness, with the attack success rate decreasing by approximately 40%. For FCNs, the two adversarial training strategies yield comparable improvements, whereas for CNNs the teacher-guided approach achieves a slightly larger reduction in attack success rate than the self-adaptive adversarial training (41% versus 36%). In contrast, LSTM-based models show a more moderate improvement, with robustness gains that remain limited compared to those observed for FCNs and CNNs. In this case, the self-adaptive strategy provides a more pronounced benefit than the teacher-guided approach, with ASR dropping by 26% versus 11%.

The interpretation of white-box (WB) robustness requires a more nuanced analysis. For CNN and LSTM architectures, adversarial training leads to a substantial improvement in white-box robustness, with reduction in ASR on the order of 50% when compared to the standard baseline. For CNNs, the self-adaptive adversarial training achieves the highest robustness, whereas for LSTMs the teacher-guided strategy proves more effective.

Conversely, FCN-based models exhibit an apparent degradation in white-box robustness when adversarial training is applied. In particular, the teacher-guided approach shows a relative increase in ASR of approximately 100%, while the self-adaptive approach reaches values up to 133% worse than the standard baseline. However, this behavior is largely an artifact of the evaluation protocol and the aggregation procedure, rather than a genuine loss of robustness.

A first contributing factor is the interaction between baseline detection performance and adversarial sample generation. In the adopted evaluation setup, only malicious samples that are correctly detected during the baseline evaluation are eligible for adversarial perturbation. As a result, the standard FCN, which exhibits a comparatively low true positive rate, is evaluated on a smaller and easier subset of malicious samples. In contrast, adversarially trained FCNs are tested against a broader and inherently more challenging set of instances, including samples that were already difficult to detect prior to perturbation.

A second, and more pronounced, effect emerges when attack success rate is aggregated across vehicles. As shown in Figure 5.1, for the *Impala* vehicle the standard FCN and the teacher-guided FCN do not appear in the white-box ASR analysis due to having zero true positives in the baseline evaluation. Conversely, the FCN trained with the self-adaptive strategy achieves a non-zero baseline performance on the same vehicle (with an F1-score of approximately 0.39), which then results in a white-box ASR value close to one. When

aggregated across vehicles, this isolated but extreme value disproportionately increases the average white-box attack success rate of the self-adaptive-trained FCN, further amplifying the apparent gap with respect to the other FCN variants.

Therefore, the observed degradation in FCN white-box robustness should not be interpreted as evidence of adversarial training being detrimental. Instead, it highlights the sensitivity of white-box robustness metrics to baseline detection performance and vehicle-level data sparsity. For CNN and LSTM architectures, whose baseline detection capabilities are consistently higher across vehicles, these effects are significantly attenuated, resulting in a more faithful reflection of the actual robustness improvements introduced by adversarial training.

We now turn to the analysis of the remaining aggregate metrics supporting robustness, namely the required perturbation magnitude, the number of optimization steps, and adversarial transferability.

Optimization steps With respect to the number of optimization steps, both in white-box and grey-box settings, the observed behavior does not exhibit a clear or consistent monotonic trend across architectures and training strategies. Intuitively, increased robustness would be expected to correlate with a higher number of attack iterations required to successfully evade the IDS. However, the experimental results reveal a highly variable behavior after adversarial training, with the required number of optimization steps showing neither a stable increase nor a systematic decrease when compared to non-adversarially trained models.

Perturbation magnitude Perturbation magnitude exhibits a partially different behavior across threat models. In the grey-box setting, adversarial training is generally associated with a slight increase in the required perturbation magnitude, suggesting a general improvement. In contrast, in the white-box setting, perturbation magnitude follows a pattern similar to that observed for optimization steps, with no clear or uniform trend emerging across models and training configurations.

This lack of a clear trend is further highlighted by the per-vehicle visualizations reported in Figures 5.2, 5.3 and 5.6, where both the number of optimization steps and the white-box perturbation magnitude display heterogeneous and vehicle-dependent patterns.

Overall, this behavior suggests that adversarially trained models learn decision boundaries that are more effective at rejecting a broader class of attacks, but primarily for perturbations exceeding a certain magnitude. For smaller perturbations, residual vul-

nerabilities persist, allowing attacks to succeed with relatively limited effort. In other words, adversarial training improves robustness in a non-uniform manner across the perturbation spectrum, strengthening resistance to stronger attacks while leaving room for improvement against weaker ones.

This observation highlights a natural direction for further optimization. In this work, the perturbation budget and the number of attack iterations used during adversarial training were selected to preserve continuity with the original CANEDERLI methodology [4]. Fine-tuning these hyperparameters, most notably by reducing the perturbation budget and/or increasing the number of attack steps during training, has strong potential to further enhance the robustness of both adversarial training strategies considered in this thesis.

Adversarial transferability Adversarial transferability exhibits a more predictable trend. As robustness increases, adversarial examples generated on one model tend to transfer more effectively to other architectures. This effect is particularly pronounced for the self-adaptive adversarial training method, which achieves substantially higher transferability gains, reaching values close to +90% in the case of CNN-based models.

Although transferability is not a primary robustness indicator in the evaluation setting adopted in this work, it provides useful insight into the structural nature of learned vulnerabilities. High transferability indicates that weaknesses exploited by adversarial examples are shared across different models, suggesting the presence of common decision boundary characteristics rather than architecture-specific flaws.

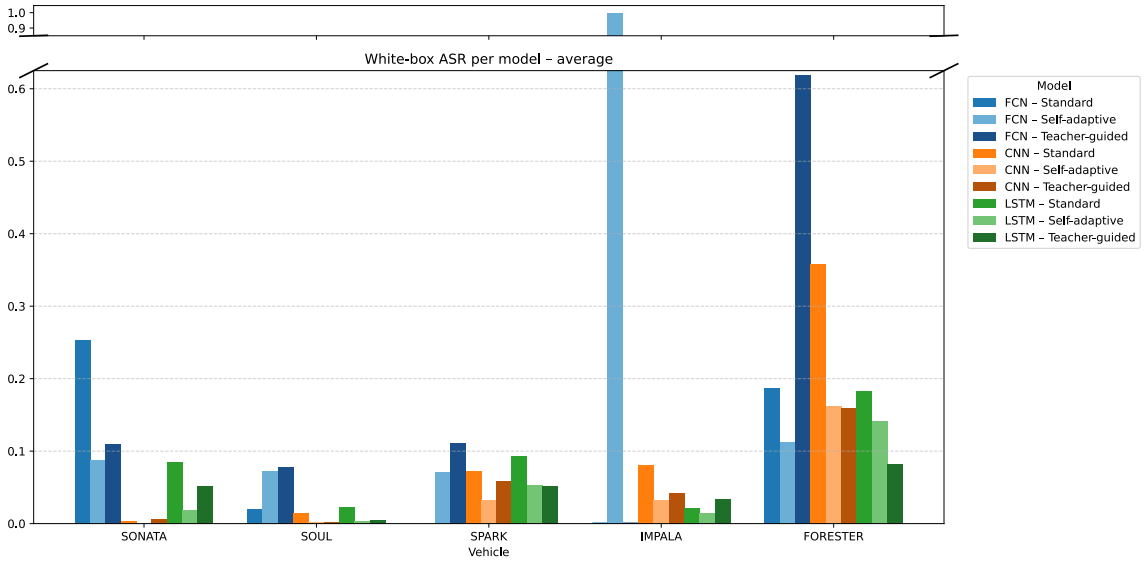


Figure 5.1: White-box attack success rate average

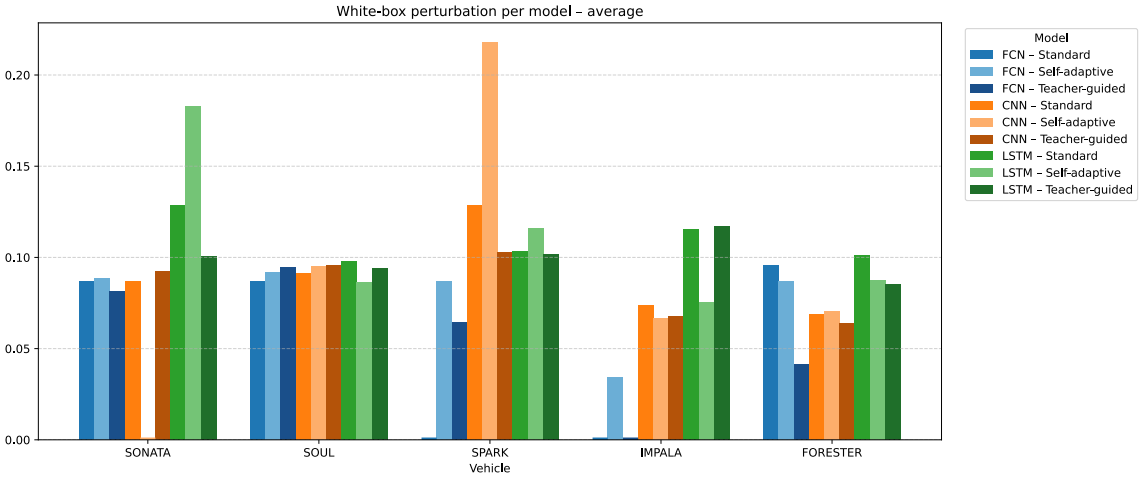


Figure 5.2: White-box perturbation average

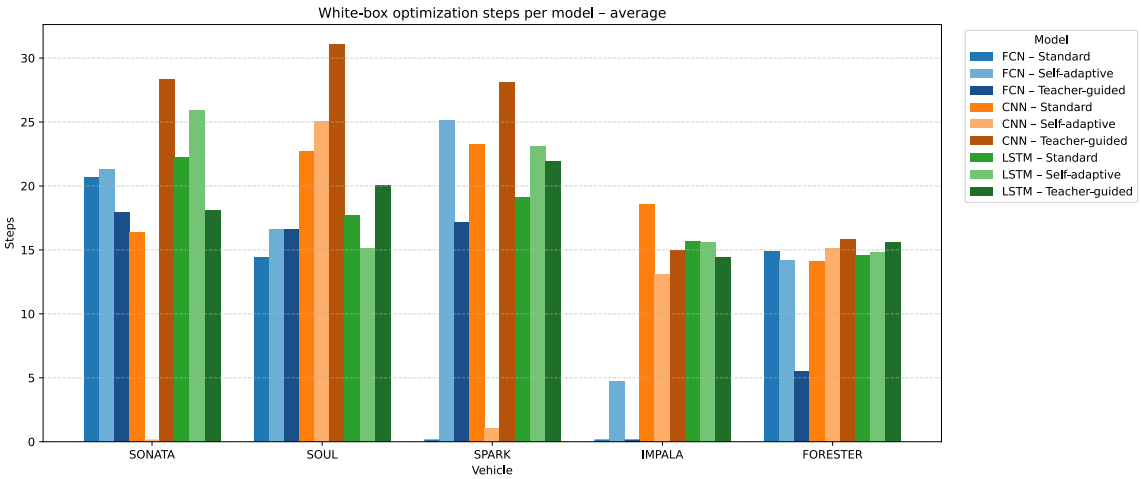


Figure 5.3: White-box optimization steps average

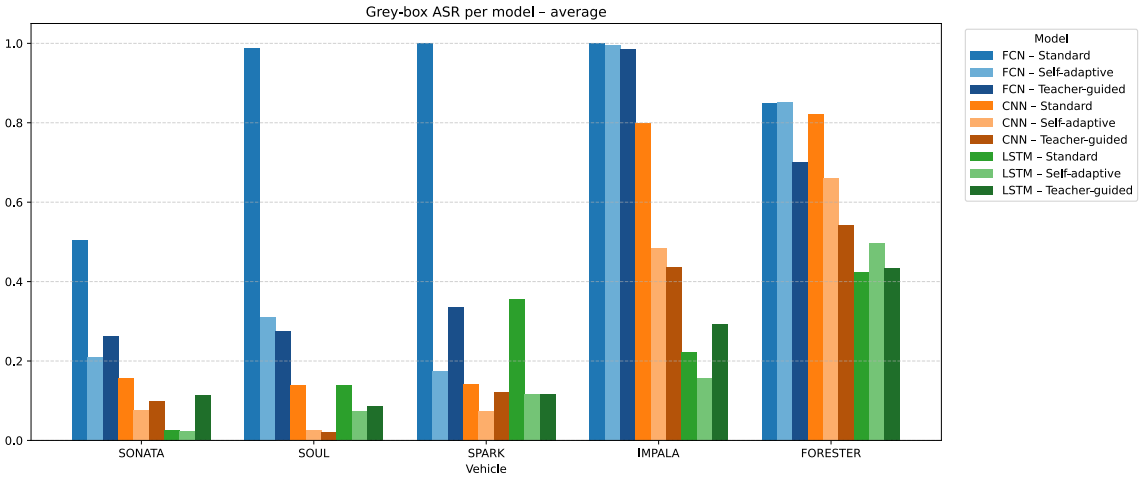


Figure 5.4: Grey-box attack success rate average

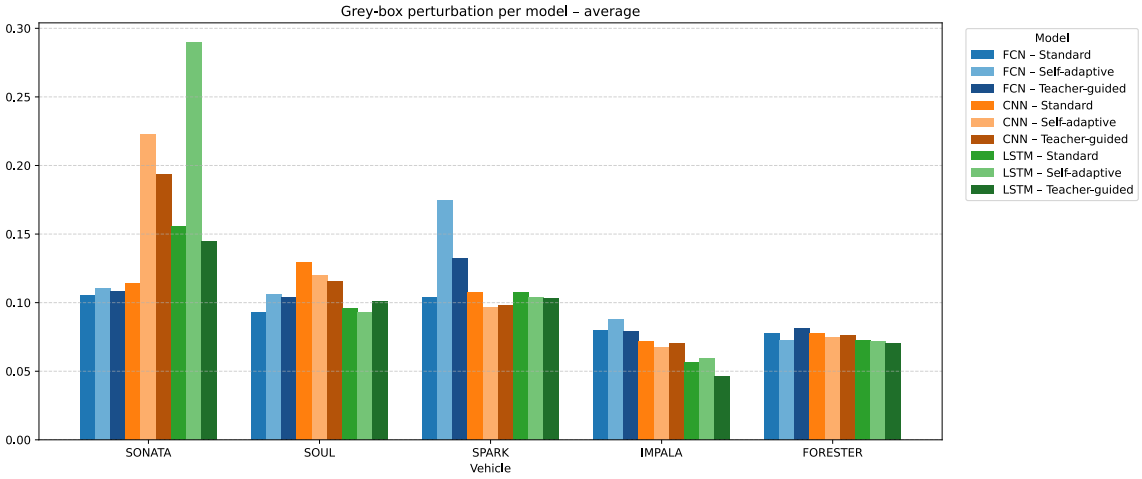


Figure 5.5: Grey-box perturbation average

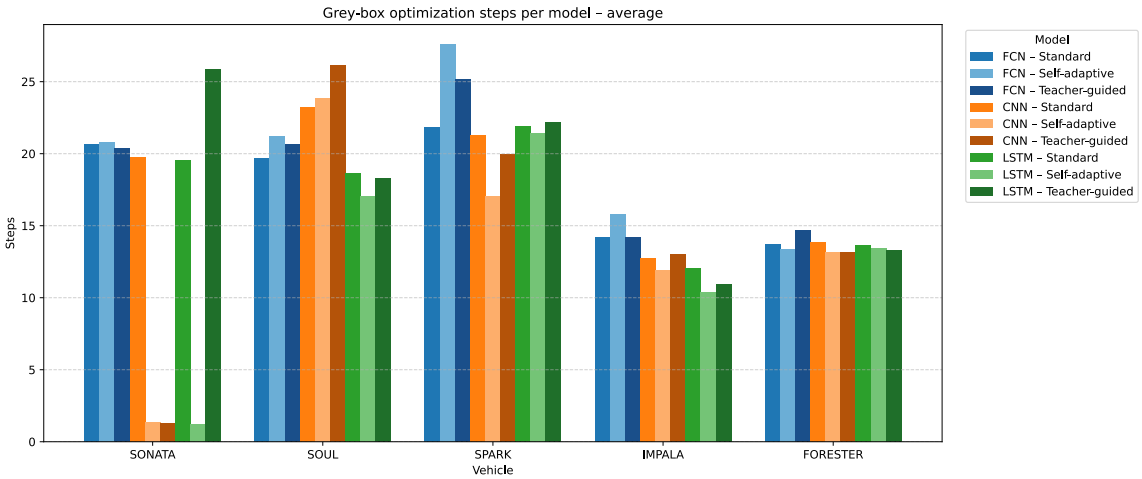


Figure 5.6: Grey-box optimization steps average

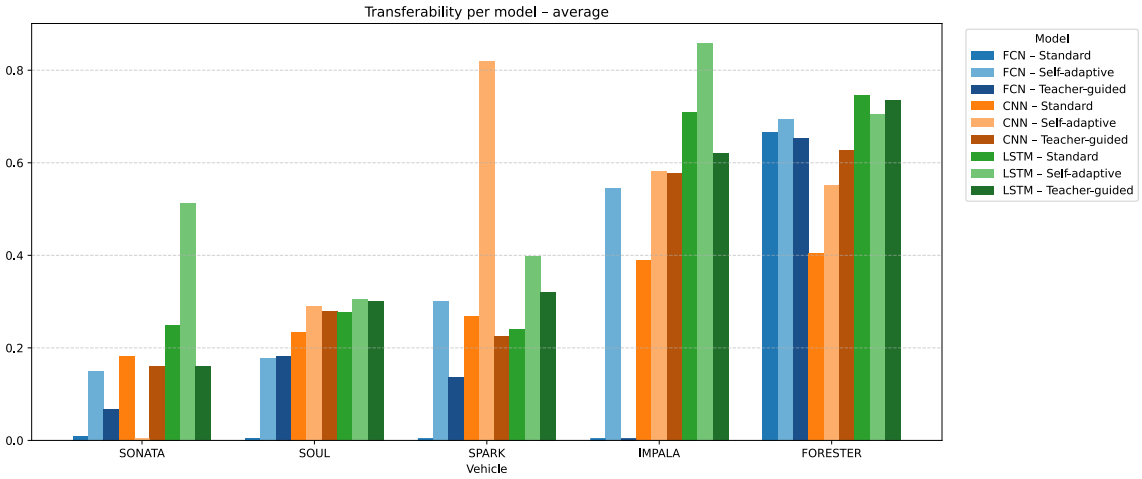


Figure 5.7: Transferability average

Shifting the analysis to a finer level of granularity by examining the aggregated results at the level of individual vehicles, allows us to highlight architecture and dataset dependent behaviors that are partially masked when considering global averages.

A singularity emerges for the teacher-guided FCN architecture. When evaluated on the *Forester* dataset, this model exhibits increased sensitivity to direct gradient-based attacks in the white-box setting. However, in the grey-box scenario, it shows a markedly improved robustness with respect to adversarial examples generated by other models based on the same architecture.

Such behavior indicates that, although the teacher-guided FCN remains vulnerable when the attacker has full access to its parameters, its learned decision boundaries are more effective at rejecting transferred adversarial samples. As a consequence, attacks crafted on surrogate models have a lower success rate when applied to the teacher-guided FCN, resulting in a tangible improvement in grey-box robustness.

This finding highlights an important distinction between white-box robustness and robustness under more realistic threat assumptions. In practical deployment scenarios, where attackers typically rely on surrogate models rather than exact knowledge of the target IDS, the teacher-guided FCN provides stronger protection than other FCN-based variants, despite its comparatively weaker white-box performance.

A particularly relevant pattern emerges for vehicles in which the FCN architecture already exhibits strong baseline detection performance. In these cases, the robustness gains induced by adversarial training are substantially larger than those observed in the aggregate analysis. This effect is especially evident for the *Sonata* and *Soul* datasets (Fig 5.4).

For both vehicles, the baseline FCN model is almost entirely unable to recognize adversarial samples in the grey-box setting, yielding attack success rate values close to one (GB ASR = 0.9999 for Sonata and 0.9888 for Soul). Using adversarial training, this behavior changes dramatically: the resulting FCN models achieve ASR levels that are significantly closer to those of more complex architectures, such as CNNs and LSTMs.

This observation suggests that, when the underlying feature representation is sufficiently informative and the baseline classifier already captures meaningful decision boundaries, adversarial training can be particularly effective even for simple, stateless architectures. In these scenarios, adversarial training acts as a powerful regularizer that sharpens existing decision boundaries rather than compensating for representational deficiencies.

> *Overall, the adversarial evaluation results confirm that both adversarial training strategies considered in this work substantially improve the robustness of CAN-based IDSs against gradient-based evasion attacks. Across architectures and vehicles, adversarially trained models consistently exhibit lower attack success rates in both white-box and grey-box settings.*

At the same time, the analysis highlights that robustness improvements are not uniform across all metrics: while attack success rates decrease significantly, the corresponding trends in perturbation magnitude and optimization steps reveal residual vulnerabilities for low-budget or low-iteration attacks, especially in the white-box scenario. These observations provide important insights into the nature of robustness induced by adversarial training. Rather than uniformly hardening the decision boundary, adversarial training primarily improves resilience against stronger perturbations, while leaving some sensitivity to minimal perturbations unresolved.

> *This suggests that the robustness gains reported in this work should be interpreted as a lower bound on the achievable performance, and that additional fine-tuning of training hyperparameters, such as perturbation budgets, attack diversity, and iteration counts, represents a promising direction for further strengthening IDS defenses.*

> *Taken together, these results demonstrate that adversarial training can be effectively integrated into CAN-based IDS pipelines to substantially enhance robustness without sacrificing baseline detection performance. Moreover, the consistency of the observed trends across architectures, vehicles, and threat models provides strong empirical evidence that adversarial training is not merely a defensive add-on, but a fundamental component for building resilient automotive intrusion detection systems.*

Additional detailed results. For completeness and to ensure full transparency of the experimental analysis, finer-grained tables reporting the complete adversarial robustness results are provided in Appendix A. These tables include per-vehicle and per-attack breakdowns of the evaluated metrics, complementing the aggregated results discussed throughout this chapter and enabling a more detailed inspection of architecture- and dataset-specific behaviors.

6 | Limitations

6.1. Discussion of the main limitations affecting the obtained results

Despite the methodological rigor and the effort to ensure fair and unbiased comparisons, several intrinsic limitations have influenced the experimental outcomes of this work. The main factors affecting the validity and generalizability of the obtained results lies in the characteristics of the datasets used throughout all experiments.

As discussed in Section 5.3, these datasets represents two of the most comprehensive and high-quality open-source resources currently available for CAN intrusion detection research. Nevertheless, their structure introduces a number of constraints that may have indirectly influenced the comparative outcomes across models and training strategies [27][28][29].

Limited data volume and class diversity A first limitation concerns the overall data volume and class distribution of the `Survival` dataset. Although the dataset includes multiple vehicles and attack types, the total number of samples per vehicle and per attack class remains relatively limited when compared to more recent large-scale CAN datasets. This constraint is particularly relevant for data-hungry architectures, such as CNNs and LSTMs, whose performance and robustness properties tend to scale with the availability of diverse training samples.

Concretely, the Sonata, Soul, and Spark datasets contain approximately 535 041, 707 843, and 402 056 total samples, respectively. By contrast, more recent benchmarks, such as the Impala dataset included in the `CAN-train-and-test` suite, provide over 14 million attack-free samples alone [24]. The relatively smaller sample size of Survival may therefore limit the representativeness of the learned decision boundaries and amplify variance across vehicles and attack classes.

This limitation is particularly relevant when interpreting adversarial robustness results, as

adversarial training implicitly relies on sufficient coverage of the input space to generalize robustness beyond the specific perturbations observed during training. As a consequence, some of the observed robustness gains may be conservative estimates of what could be achieved on larger and more diverse datasets.

Simplified attack implementations A second limitation arises from the nature of the attack implementations included in the `Survival` dataset. While the provided attacks are well-defined and reproducible, they are relatively simplistic. As highlighted by Verma et al. [28], such attacks may not fully capture the sophistication, adaptability, and stealth characteristics of modern real-world adversaries.

In particular, the absence of multi-stage attacks, adaptive strategies, and context-aware manipulations limits the diversity of adversarial behaviors to which the IDS models are exposed.

Localized temporal concentration of malicious traffic A further limitation concerns the temporal distribution of malicious packets within the `CAN-train-and-test` traces. In most scenarios, attacks are confined to short and contiguous segments of the logs, while the remaining portions consist almost exclusively of benign traffic. This uneven temporal concentration complicates dataset partitioning, as naive or random splits can easily introduce severe class imbalance or unintended data leakage between training and validation subsets.

These structural constraints reduce the flexibility in constructing balanced and fully independent datasets, thereby potentially affecting both training stability and the reliability of comparative evaluations across models and training strategies.

Limited malicious payload diversity In the `CAN-train-and-test` dataset, for each vehicle and attack scenario, only a small number of distinct malicious payloads are available (typically two for training and two for testing), while the remaining malicious samples consist of repeated transmissions of the same payloads with varying inter-arrival times. This structural constraint inherently limits the ability of supervised models to learn truly generalizable decision boundaries. Instead of identifying attacks based on semantic features or contextual dynamics, models may partially overfit to the specific payload patterns observed during training.

As a consequence, certain architectures or training strategies may appear more effective not because of superior robustness or generalization capabilities, but because their learned representations align more closely with these repetitive payload structures.

This limited diversity of malicious payloads manifests regardless of the adopted splitting strategy.

When payload disjointness is enforced, models are evaluated on malicious samples that are syntactically unseen, but learned from a narrow and weakly diversified training experience. In this case, robustness and detection performance are likely to be underestimated, as models are required to generalize from a limited set of attack patterns.

Conversely, when payload disjointness is not enforced, models are exposed to a larger number of distinct malicious samples during training, but evaluation is performed on payloads that may have already been observed. This setting can lead to an overestimation of robustness and detection capabilities, as the test phase partially reflects memorization rather than generalization.

In this work, the former strategy is deliberately adopted, as it aligns with established best practices for supervised learning evaluation and avoids optimistic bias, despite its tendency to yield conservative performance estimates.

Absence of black-box adversarial evaluations. A further limitation concerns the scope of the adversarial robustness assessment. Given the already limited data volume and attack complexity of the `Survival` dataset, together with the restricted diversity of malicious payloads in the `CAN-train-and-test` dataset, adopting a fully black-box evaluation setting would have further fragmented the available data. This would have reduced the amount of supervision per attack configuration, ultimately leading to less reliable and potentially non-representative results.

For this reason, all experiments were conducted under white-box and grey-box threat models, as defined in Section 5.4.1. Although this methodological choice is consistent with prior work, most notably *Longari et al.*[18], which reports only marginal differences between grey-box and black-box adversarial scenarios in comparable settings, it nonetheless limits the completeness of the robustness analysis.

Overall, these limitations do not undermine the validity of the experimental findings but highlight the inherent challenges of conducting reproducible and unbiased research in the automotive intrusion detection domain. They also emphasize the need for richer, more diverse datasets to enable deeper and more comprehensive evaluations.

6.2. Explanation of why these limitations could not be overcome within the current work

The limitations discussed in the previous section primarily arise from structural and practical constraints that could not be reasonably overcome within the scope of this thesis. In particular:

- the low variability of malicious payloads in the `can-train-and-test` dataset is an inherent consequence of its data collection methodology. Each attack trace was generated on real vehicles using a small, predefined set of malicious payloads, chosen to ensure operational safety, repeatability, and compliance with experimental constraints. Expanding the payload space would require the design and validation of additional attack patterns directly on physical vehicles, a process that is both costly and constrained by safety considerations.
- the limited data volume of the `Survival` dataset is similarly rooted in the complexity and expense of collecting real CAN traffic under controlled attack scenarios. Recording large-scale, high-fidelity CAN traces across multiple vehicles and attack types requires prolonged access to in-vehicle networks, specialized hardware, and carefully supervised experimental conditions.
- the simplicity of the implemented attack strategies in the `Survival` dataset reflects the state of the art at the time of its creation and the emphasis on reproducibility rather than adversarial sophistication. Retrofitting more complex or adaptive attack behaviors into the dataset would require re-running the original data acquisition pipeline, which is not feasible without direct access to the original experimental setup.

More generally, generating additional benign and malicious samples with higher diversity, necessary to support more reliable statistical estimates and to enable fully black-box adversarial evaluations, would require direct access to in-vehicle CAN networks and dedicated hardware infrastructures. Such conditions fall outside the practical scope of this work.

As a result, the experimental analysis was necessarily constrained to the available datasets, which nevertheless remain among the most comprehensive and realistic open-source benchmarks currently available for CAN-based intrusion detection research. Despite their limitations, they provide a valuable and widely adopted reference point for comparative evaluation.

In summary, the identified constraints are the result of external and structural limitations rather than methodological oversights. The experimental design therefore reflects a deliberate balance between rigor, reproducibility, and feasibility within the boundaries imposed by publicly available resources.

7 | Future works

The experimental analysis conducted in this thesis has provided valuable insights into the adversarial robustness of CAN-based Intrusion Detection Systems. Nevertheless, several opportunities remain to extend this work and further advance the state of the art in automotive cybersecurity.

7.1. Enhanced dataset diversity and accessibility

A recurring limitation throughout this study concerns the characteristics of the datasets employed (see Chapter 6). Since the validity and generalizability of robustness evaluations are strongly dependent on the quality and diversity of the underlying data, future research should prioritize the creation and sharing of richer CAN datasets.

In particular, future efforts should focus on collecting or generating datasets that encompass a larger number of distinct and more sophisticated malicious payloads, as well as a broader range of benign driving conditions. Possible directions include:

- **Preferably, real-world data acquisition:** conducting extensive and controlled experiments on real vehicles to generate multiple variations of each attack type, thereby capturing a wider spectrum of CAN traffic behaviors and attack manifestations.
- **Alternatively, synthetic data generation:** leveraging high-fidelity simulation frameworks to safely synthesize diverse malicious payloads, potentially combined with domain adaptation or augmentation techniques to preserve the statistical and temporal properties of real CAN traffic.

Beyond data diversity, an important yet often overlooked aspect concerns dataset accessibility and usability. Publishing datasets in standardized, well-documented, and analysis-ready formats would significantly lower the entry barrier for reproducibility and comparative evaluation, fostering more systematic and large-scale studies on CAN-based intrusion detection and adversarial robustness.

7.2. Black-box adversarial evaluations

Due to the structural constraints of the available datasets, the present study focuses exclusively on white-box and grey-box adversarial threat models. Future work should extend the experimental evaluation to fully black-box scenarios in order to assess robustness and adversarial transferability under more realistic attacker assumptions, where neither model parameters nor architectures are known. Such evaluations would provide a more comprehensive understanding of IDS resilience in practical deployment settings.

7.3. Optimization of adversarial training hyperparameters

As discussed in Chapter 5, the hyperparameters adopted for adversarial training in this work closely follow those of the original CANEDERLI methodology [4], in order to ensure methodological continuity. In particular, both the perturbation budget and the number of attack iterations employed during adversarial sample generation are kept aligned with the reference implementation.

The experimental results suggest that adversarial training improves robustness in a non-uniform manner across the perturbation spectrum, primarily strengthening resistance against higher-magnitude attacks while leaving residual vulnerabilities for low-budget perturbations.

Future research should therefore systematically investigate the impact of adversarial training hyperparameters on robustness outcomes. Key directions include:

- **Perturbation budget tuning:** exploring the effect of smaller perturbation budgets during training, which may encourage the model to develop tighter and more locally stable decision boundaries.
- **Increasing attack iterations:** analyzing whether a higher number of optimization steps during adversarial sample generation leads to more challenging training examples and, consequently, stronger robustness.

A systematic hyperparameter exploration could reveal robustness-accuracy trade-offs that were not observable under the configuration adopted in this thesis. Such an analysis would provide a deeper understanding of how adversarial training shapes decision boundaries in the CAN domain and could lead to IDS models that are robust not only under fixed evaluation settings, but across a broader range of adversarial conditions.

7.4. Architectural and defensive enhancements

The results presented in this thesis highlight the complexity of the interaction between model architecture, training strategy, and adversarial robustness. Future research could therefore focus on analyzing complementary aspects of these components, with the goal of further strengthening CAN-based intrusion detection systems against adversarial attacks.

- **Teacher model complexity in teacher-guided adversarial training:** the proposed teacher-guided strategy opens an additional research dimension related to the role and design of the teacher itself. In this work, the teacher models were selected to ensure controlled and fair comparisons. Future studies could investigate the impact of employing more complex teacher architectures, stronger training procedures, or differently regularized models, and analyze more in detail how the quality and robustness of the teacher influence the robustness, generalization, and stability of the student.
- **Normalization and regularization techniques:** investigating the impact of normalization and regularization methods whose benefits for adversarial robustness have been demonstrated in other domains, such as computer vision, and assessing whether similar effects generalize to CAN-based intrusion detection.
- **Architectural and training combinations:** exploring alternative combinations of network architectures, feature representations, and training paradigms to identify configurations that achieve a more favorable balance between detection accuracy, robustness, and computational efficiency.
- **Hybrid defensive strategies:** evaluating the effectiveness of hybrid approaches that integrate complementary detection mechanisms, such as coupling adversarially trained classifiers with anomaly-based or generative models, in order to leverage their respective strengths and mitigate individual weaknesses.

7.5. Real-world deployment considerations

Another promising research direction concerns the assessment of IDS robustness under realistic deployment constraints. In particular, future work should consider:

- **Field testing:** conducting on-road or hardware-in-the-loop experiments to evaluate the practical impact of adversarial samples that successfully evade detection, and to validate whether robustness gains observed in controlled settings translate to real-world scenarios.

- **Real-time detection requirements:** analyzing the trade-offs between detection latency, computational overhead, and adversarial robustness to ensure reliable operation within the strict timing constraints imposed by in-vehicle networks.

7.6. Advanced adversarial attack modeling

Finally, future research could focus on the design and evaluation of more sophisticated adversarial attacks tailored to the automotive domain. This includes modeling coordinated attack strategies that combine payload manipulation with timing-based or protocol-level anomalies, with the goal of simulating realistic and adaptive adversarial behaviors within in-vehicle CAN networks.

Overall, these research directions have the potential to significantly enhance the robustness, applicability, and practical relevance of CAN-based intrusion detection systems. By bridging the gap between controlled academic benchmarks and real-world deployment conditions, future work can contribute to the development of IDS solutions that are not only accurate, but also resilient by design.

8 | Conclusions

This thesis investigated the problem of adversarial robustness in CAN-based Intrusion Detection Systems, with the objective of understanding whether adversarial training can improve resilience to evasion attacks without compromising detection performance on legitimate traffic.

The work first established a comprehensive experimental framework grounded in clearly defined threat models, carefully selected datasets, and reproducible evaluation procedures. Building upon this foundation, two adversarial training strategies were considered: a self-adaptive adversarial training approach derived from the method originally proposed in CANEDERLI, and a novel teacher-guided adversarial training strategy designed to decouple adversarial example generation from the optimization of the target model.

Extensive experiments were conducted to evaluate the impact of these strategies across multiple architectures and evaluation settings, including white-box and grey-box scenarios. The analysis considered both baseline detection performance and multiple robustness indicators, such as attack success rates, perturbation characteristics, and transferability patterns.

The results show that adversarial training can significantly improve robustness against evasion attacks while maintaining competitive detection performance on clean traffic. At the same time, the study revealed that robustness improvements are not uniform across the perturbation spectrum, highlighting residual vulnerabilities against low-magnitude perturbations and emphasizing the importance of careful hyperparameter selection.

Beyond the empirical findings, this work contributes a systematic analysis of how training strategies, dataset characteristics, and threat assumptions interact in shaping adversarial robustness in automotive intrusion detection. The discussion of limitations and future directions further outlines concrete paths toward more realistic evaluations, richer datasets, and stronger defensive mechanisms.

Overall, the results reinforce the importance of incorporating adversarial considerations into the design of machine learning-based IDSs for in-vehicle networks. As automotive sys-

tems continue to evolve toward increasingly connected and software-defined architectures, developing robust detection mechanisms will remain a critical component in ensuring the security and reliability of next-generation vehicles.

Bibliography

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[4] Francesco Marchiori and Mauro Conti. Canederli: On the impact of adversarial training and transferability on can intrusion detection systems. In *Proceedings of the 2024 ACM Workshop on Wireless Security and Machine Learning*, pages 8–13, 2024.

[5] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[6] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[9] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In

2016 IEEE European symposium on security and privacy (EuroS&P), pages 372–387. IEEE, 2016.

[10] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.

[11] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 130–139. IEEE, 2016.

[12] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. Detecting attacks on the can protocol with machine learning. *Annu EECS*, 558(7), 2016.

[13] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21: 100198, 2020.

[14] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th annual conference on privacy, security and trust (PST)*, pages 1–6. IEEE, 2018.

[15] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. Canet: An unsupervised intrusion detection system for high dimensional can bus data. *Ieee Access*, 8:58194–58205, 2020.

[16] Stefano Longari, Carlo Alberto Pozzoli, Alessandro Nichelini, Michele Carminati, and Stefano Zanero. Candito: Improving payload-based detection of attacks on controller area networks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 135–150. Springer, 2023.

[17] Stefano Longari, Francesco Noseda, Michele Carminati, and Stefano Zanero. Evaluating the robustness of automotive intrusion detection systems against evasion attacks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 337–352. Springer, 2023.

[18] Stefano Longari, Paolo Cerracchio, Michele Carminati, and Stefano Zanero. Assessing the resilience of automotive intrusion detection systems to adversarial manipulation. *ACM Transactions on Cyber-Physical Systems*, 2025.

[19] Fatimah Aloraini, Amir Javed, and Omer Rana. Adversarial attacks on intrusion detection systems in in-vehicle networks of connected and autonomous vehicles. *Sensors*, 24(12):3848, 2024.

[20] Haihang Zhao, Yi Wang, Anyu Cheng, Shanshan Wang, Jing Yuan, and Hongrong Wang. Safeguarding gru-based intrusion detection systems from adversarial attacks with dynamic label watermark in can bus communication. *IEEE Internet of Things Journal*, 2024.

[21] Junman Qin, Yijie Xun, Zhouyan Deng, and Jiajia Liu. Gpids: Gan assisted contextual pattern-aware intrusion detection system for ivn. *IEEE Transactions on Vehicular Technology*, 73(9):12682–12693, 2024.

[22] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[23] Mee Lan Han, Byung Il Kwak, and Huy Kang Kim. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular communications*, 14:52–63, 2018.

[24] Brooke Lampe and Weizhi Meng. Can-train-and-test: A curated can dataset for automotive intrusion detection. *Computers & Security*, 140:103777, 2024.

[25] Brooke Kidmose and Weizhi Meng. can-sleuth: Investigating and evaluating automotive intrusion detection datasets. In *Proceedings of the 2024 European Interdisciplinary Cybersecurity Conference*, pages 19–28, 2024.

[26] Brooke Kidmose, Andreas Kidmose, and Weizhi Meng. can-sleuth: Sleuthing out the capabilities, limitations, and performance impacts of automotive intrusion detection datasets. *International Journal of Information Security*, 24(5):193, 2025.

[27] Seyoung Lee, Wonsuk Choi, Insup Kim, Ganggyu Lee, and Dong Hoon Lee. A comprehensive analysis of datasets for automotive intrusion detection systems. *Computers, Materials & Continua*, 76(3), 2023.

[28] Miki E Verma, Robert A Bridges, Michael D Iannacone, Samuel C Hollifield, Pablo Moriano, Steven C Hespeler, Bill Kay, and Frank L Combs. A comprehensive guide to can ids data and introduction of the road dataset. *PLoS one*, 19(1):e0296879, 2024.

[29] Arash Vahidi, Thomas Rosenstatter, and Nishat I Mowla. Systematic evaluation of automotive intrusion detection datasets. In *Proceedings of the 6th ACM Computer Science in Cars Symposium*, pages 1–12, 2022.

A | Appendix A

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	0.2531	0.0871	20.72	0.5039	0.1053	20.67	0.0098
	self-adaptive	0.087	0.0887	21.30	0.2111	0.1107	20.83	0.1488
	teacher-guided	0.1103	0.0812	17.94	0.2622	0.1085	20.41	0.0671
CNN	standard	0.0031	0.087	16.41	0.1563	0.1143	19.76	0.1813
	self-adaptive	0	-	-	0.0765	0.2228	1.32	-
	teacher-guided	0.006	0.0925	28.36	0.0996	0.1936	1.3	0.1616
LSTM	standard	0.0854	0.1287	22.23	0.0261	0.1557	19.56	0.2502
	self-adaptive	0.0182	0.1829	25.91	0.0227	0.2895	1.23	0.5119
	teacher-guided	0.0514	0.1004	18.11	0.113	0.1449	25.88	0.1597

Table A.1: Mean adversarial performance on perturbed CAN traffic - Hyundai Sonata

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	0.0198	0.0868	14.41	0.9888	0.0926	19.68	0.0052
	self-adaptive	0.0727	0.092	16.58	0.3107	0.1062	21.18	0.1783
	teacher-guided	0.0784	0.0943	16.62	0.2744	0.1037	20.66	0.1823
CNN	standard	0.0142	0.0913	22.71	0.1385	0.1292	23.25	0.234
	self-adaptive	0.0026	0.0949	25.03	0.0259	0.1197	23.86	0.2902
	teacher-guided	0.0015	0.0958	31.07	0.0202	0.1153	26.13	0.2785
LSTM	standard	0.0231	0.098	17.74	0.1384	0.0958	18.61	0.2765
	self-adaptive	0.0037	0.0865	15.11	0.0744	0.0929	17.08	0.3058
	teacher-guided	0.005	0.094	20.03	0.0867	0.1009	18.28	0.3005

Table A.2: Mean adversarial performance on perturbed CAN traffic - Kia Soul

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	0	-	-	0.9999	0.1036	21.82	-
	self-adaptive	0.0708	0.0869	25.11	0.1745	0.1749	27.59	0.3021
	teacher-guided	0.111	0.0643	17.18	0.336	0.1323	25.18	0.1376
CNN	standard	0.072	0.1288	23.24	0.1422	0.1076	21.31	0.268
	self-adaptive	0.0323	0.2177	1.05	0.0727	0.0969	17.07	0.8206
	teacher-guided	0.059	0.103	28.09	0.1227	0.0982	19.95	0.2259
LSTM	standard	0.0937	0.1032	19.1	0.3553	0.1074	21.93	0.2401
	self-adaptive	0.0528	0.1159	23.09	0.1178	0.1042	21.44	0.3982
	teacher-guided	0.0523	0.1019	21.95	0.1161	0.1028	22.16	0.3209

Table A.3: Mean adversarial performance on perturbed CAN traffic - Chevrolet Spark

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	-	-	-	1	0.0801	14.2	-
	self-adaptive	0.9962	0.034	4.72	0.9967	0.0879	15.82	0.5443
	teacher-guided	-	-	-	0.9863	0.0789	14.2	-
CNN	standard	0.0812	0.074	18.54	0.7999	0.072	12.75	0.3887
	self-adaptive	0.0329	0.0666	13.12	0.4834	0.0671	11.89	0.582
	teacher-guided	0.0421	0.0676	14.97	0.4364	0.0702	13.04	0.5771
LSTM	standard	0.0212	0.1152	15.66	0.2234	0.0564	12.08	0.7091
	self-adaptive	0.0149	0.0755	15.61	0.1572	0.0597	10.37	0.8581
	teacher-guided	0.0337	0.1171	14.43	0.2928	0.0463	10.94	0.6211

Table A.4: Mean adversarial performance on perturbed CAN traffic - Chevrolet Impala

model	method	WB ASR	WB perturbation	WB steps	GB ASR	GB perturbation	GB steps	transferability
FCN	standard	0.1863	0.0954	14.86	0.8506	0.078	13.69	0.6673
	self-adaptive	0.1128	0.0869	14.23	0.8515	0.0725	13.4	0.6951
	teacher-guided	0.6188	0.0415	5.48	0.7017	0.0815	14.7	0.6528
CNN	standard	0.3581	0.0686	14.13	0.8225	0.0777	13.83	0.4047
	self-adaptive	0.1622	0.0705	15.15	0.6604	0.0749	13.13	0.5508
	teacher-guided	0.1587	0.0637	15.8	0.5433	0.076	13.13	0.6274
LSTM	standard	0.1834	0.1014	14.6	0.4248	0.0726	13.66	0.7469
	self-adaptive	0.1416	0.0873	14.83	0.4965	0.072	13.4	0.7044
	teacher-guided	0.0817	0.0855	15.67	0.4332	0.0704	13.27	0.7351

Table A.5: Mean adversarial performance on perturbed CAN traffic - Subaru Forester

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.7143	0.4596	0.375	-	0.25	0.3333	-	0.3242
	self-adaptive	0.0234	1.0	0.2584	0.0	-	0.0	0.2333	-	0.0934
	teacher-guided	0.0321	0.4941	1.0	0.0625	-	0.0	0.2333	-	0.1154
CNN	standard	0.0	0.0	0.0	1.0	-	1.0	0.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	1.0	0.0	-	0.0
LSTM	standard	0.0	0.0	0.0	0.0	-	0.0	1.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	0.0	0.0	-	1.0

Table A.6: ASR matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0537	0.0483	0.0494	0.0467	-	0.063	0.0661	-	0.0534
	self-adaptive	0.0518	0.0495	0.0513	-	-	-	0.0785	-	0.0529
	teacher-guided	0.0439	0.0504	0.051	0.051	-	-	0.066	-	0.0507
CNN	standard	-	-	-	0.0456	-	0.0607	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	0.0607	-	-	-
LSTM	standard	-	-	-	-	-	-	0.0553	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	-	-	-	0.0503

Table A.7: Perturbation matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	21.28	13.6	13.81	12.33	-	17.0	19.7	-	14.92
	self-adaptive	13.87	14.15	14.33	-	-	-	23.0	-	14.29
	teacher-guided	11.69	14.45	14.32	13.0	-	-	20.0	-	13.76
CNN	standard	-	-	-	12.0	-	17.0	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	17.0	-	-	-
LSTM	standard	-	-	-	-	-	-	16.2	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	-	-	-	13.64

Table A.8: Steps matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.8969	0.1394	0.6176	-	0.4565	0.3947	0.6923	0.3215
	self-adaptive	0.019	1.0	0.0759	0.2255	-	0.2681	0.2303	0.3846	0.0887
	teacher-guided	0.0273	0.1522	1.0	0.2059	-	0.3225	0.3026	0.4615	0.1175
CNN	standard	0.0015	0.0	0.0	1.0	-	0.0833	0.0	0.0	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	1.0	0.0	0.0	0.0
LSTM	standard	0.0015	0.0	0.0	0.1111	-	0.0	1.0	0.0	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	1.0	0.0
	teacher-guided	0.0046	0.0	0.0	0.1111	-	0.25	0.1667	0.0	1.0

Table A.9: ASR matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0856	0.0989	0.0848	0.1436	-	0.1358	0.1151	0.1464	0.097
	self-adaptive	0.082	0.0987	0.0818	0.1483	-	0.1352	0.1287	0.144	0.0877
	teacher-guided	0.0779	0.0906	0.077	0.153	-	0.1333	0.1251	0.1486	0.0915
CNN	standard	0.1326	-	-	0.1385	-	0.0793	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	0.1361	-	-	-
LSTM	standard	0.1144	-	-	0.1524	-	-	0.1143	-	-
	self-adaptive	-	-	-	-	-	-	-	0.1458	-
	teacher-guided	0.1033	-	-	0.1701	-	0.1518	0.1623	-	0.0949

Table A.10: Perturbation matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	29.25	41.37	24.5	43.06	-	40.98	35.3	50.0	28.04
	self-adaptive	22.49	40.11	23.92	46.04	-	41.74	37.63	50.0	26.73
	teacher-guided	21.72	27.0	29.34	44.95	-	40.96	37.5	50.0	28.79
CNN	standard	36.75	-	-	40.55	-	22.0	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	40.61	-	-	-
LSTM	standard	31.12	-	-	44.0	-	-	33.76	-	-
	self-adaptive	-	-	-	-	-	-	-	50.0	-
	teacher-guided	27.88	-	-	49.0	-	43.83	46.04	-	27.86

Table A.11: Steps matrix for the Sonata dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.7788	0.487	0.8571	-	-	0.8758	0.8838	0.7619
	self-adaptive	0.0777	1.0	0.3116	0.5714	-	-	0.8395	0.8187	0.6714
	teacher-guided	0.0715	0.4678	1.0	0.4286	-	-	0.8479	0.8081	0.6799
CNN	standard	0.0015	0.0023	0.0	1.0	-	-	0.9034	0.8586	0.5632
	self-adaptive	0.0	0.0	0.0	0.5	-	-	0.8022	0.798	0.5789
	teacher-guided	0.0015	0.0023	0.0	1.0	-	-	0.8652	0.8788	0.7368
LSTM	standard	0.0015	0.0023	0.0033	0.0	-	-	1.0	0.6263	0.0526
	self-adaptive	0.0	0.0	0.0	0.0	-	-	0.5281	1.0	0.1526
	teacher-guided	0.0182	0.014	0.0066	1.0	-	-	0.7708	0.9798	1.0

Table A.12: ASR matrix for the Sonata dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.1412	0.1313	0.1197	0.0968	-	-	0.3006	0.2298	0.2266
	self-adaptive	0.1966	0.1289	0.1208	0.1143	-	-	0.3055	0.2341	0.2391
	teacher-guided	0.1577	0.1337	0.1146	0.1082	-	-	0.3056	0.2347	0.2342
CNN	standard	0.1801	0.132	-	0.0884	-	-	0.3028	0.2358	0.2368
	self-adaptive	-	-	-	0.1111	-	-	0.3056	0.2385	0.2362
	teacher-guided	0.1801	0.132	-	0.0884	-	-	0.3026	0.2338	0.2248
LSTM	standard	0.17	0.1686	0.098	-	-	-	0.2911	0.2145	0.2387
	self-adaptive	-	-	-	-	-	-	0.3057	0.2199	0.2733
	teacher-guided	0.1787	0.1395	0.1395	0.0884	-	-	0.2972	0.2233	0.1918

Table A.13: Perturbation matrix for the Sonata dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.01	2.0	1.75	1.0	-	-	1.0	1.83	1.45
	self-adaptive	1.01	1.97	1.67	1.0	-	-	1.0	1.85	1.46
	teacher-guided	1.01	1.68	1.73	1.0	-	-	1.0	1.84	1.45
CNN	standard	1.0	1.5	-	1.0	-	-	1.0	1.86	1.42
	self-adaptive	-	-	-	1.0	-	-	1.0	1.88	1.41
	teacher-guided	1.0	1.5	-	1.0	-	-	1.0	1.85	1.45
LSTM	standard	1.0	2.0	1.4	-	-	-	1.0	1.62	1.41
	self-adaptive	-	-	-	-	-	-	1.0	1.82	1.47
	teacher-guided	1.02	1.64	1.83	1.0	-	-	1.0	1.83	1.44

Table A.14: Steps matrix for the Sonata dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.6929	0.4728	0.375	-	0.25	0.3333	-	0.3187
	self-adaptive	0.0246	1.0	0.2676	0.0	-	0.0	0.2667	-	0.1209
	teacher-guided	0.0314	0.5238	1.0	0.0625	-	0.0	0.2667	-	0.1319
CNN	standard	0.0	0.0	0.0	1.0	-	1.0	0.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	1.0	0.0	-	0.0
LSTM	standard	0.0	0.0	0.0	0.0	-	0.0	1.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	0.0	0.0	-	1.0

Table A.15: ASR matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0565	0.0603	0.061	0.0624	-	0.064	0.0766	-	0.0639
	self-adaptive	0.0603	0.0608	0.0602	-	-	-	0.0781	-	0.0633
	teacher-guided	0.0565	0.0615	0.0608	0.0877	-	-	0.0731	-	0.0629
CNN	standard	-	-	-	0.059	-	0.0632	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	0.0632	-	-	-
LSTM	standard	-	-	-	-	-	-	0.0712	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	-	-	-	0.0606

Table A.16: Perturbation matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	25.28	17.74	18.2	16.33	-	19.0	34.7	-	21.12
	self-adaptive	17.71	18.35	17.03	-	-	-	36.75	-	20.32
	teacher-guided	14.48	18.53	18.37	20.0	-	-	31.62	-	20.96
CNN	standard	-	-	-	14.5	-	19.0	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	19.0	-	-	-
LSTM	standard	-	-	-	-	-	-	27.87	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	-	-	-	18.88

Table A.17: Steps matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.8012	0.1505	0.5	-	0.4308	0.3725	-	0.3035
	self-adaptive	0.0237	1.0	0.0523	0.0625	-	0.1846	0.1961	-	0.1272
	teacher-guided	0.0279	0.411	1.0	0.1875	-	0.2462	0.2451	-	0.1272
CNN	standard	0.0016	0.0	0.0	1.0	-	0.2857	0.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0	0.0	0.0	0.0	-	1.0	0.0	-	0.0
LSTM	standard	0.0	0.0	0.0	0.0	-	0.0	1.0	-	0.0
	self-adaptive	0.0	0.0	0.0	0.0	-	0.0	0.0	-	0.0
	teacher-guided	0.0016	0.0	0.0	0.0	-	0.1429	0.0	-	1.0

Table A.18: ASR matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0984	0.103	0.0927	0.1103	-	0.1091	0.112	-	0.1075
	self-adaptive	0.0994	0.1055	0.1013	0.1407	-	0.1089	0.1167	-	0.1057
	teacher-guided	0.099	0.1067	0.1028	0.1206	-	0.1075	0.1201	-	0.101
CNN	standard	0.1271	-	-	0.1033	-	0.1209	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	0.1101	-	-	-
LSTM	standard	-	-	-	-	-	-	0.1115	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	0.1016	-	-	-	-	0.1182	-	-	0.1044

Table A.19: Perturbation matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	26.76	31.83	18.95	16.25	-	38.29	33.74	-	27.92
	self-adaptive	19.36	31.91	18.98	23.0	-	42.25	36.95	-	27.25
	teacher-guided	18.03	30.79	25.92	23.0	-	41.62	33.64	-	27.84
CNN	standard	34.67	-	-	14.0	-	29.5	-	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	-	-	-	36.81	-	-	-
LSTM	standard	-	-	-	-	-	-	32.31	-	-
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	28.11	-	-	-	-	41.0	-	-	28.72

Table A.20: Steps matrix for the Sonata dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.9824	0.9928	1.0	1.0	1.0	0.9697	1.0	1.0
	self-adaptive	0.0124	1.0	0.4317	0.24	0.4167	0.5	0.3333	0.1579	0.2564
	teacher-guided	0.0062	0.4075	1.0	0.26	0.4167	0.25	0.2727	0.1053	0.2051
CNN	standard	0.0	0.0	0.0017	1.0	0.0	0.0	0.0	0.0	0.0
	self-adaptive	0.0	0.0	0.0	0.1	1.0	0.0	0.0	0.0	0.0
	teacher-guided	0.0	0.0	0.0	0.1	0.0	1.0	0.0	0.0	0.0
LSTM	standard	0.0067	0.0053	0.0034	0.0	0.0	0.0	1.0	0.3333	0.3333
	self-adaptive	0.0	0.0035	0.0034	0.0	0.0	0.0	0.0833	1.0	0.3333
	teacher-guided	0.0	0.0018	0.0017	0.0	0.0	0.0	0.1667	0.3333	1.0

Table A.21: ASR matrix for the Soul dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0452	0.049	0.0503	0.0632	0.0686	0.0675	0.0424	0.0483	0.0569
	self-adaptive	0.065	0.0493	0.0497	0.068	0.0672	0.0645	0.0454	0.0409	0.0585
	teacher-guided	0.0667	0.0477	0.0504	0.0695	0.0672	0.0559	0.046	0.0392	0.0621
CNN	standard	-	-	0.076	0.0616	-	-	-	-	-
	self-adaptive	-	-	-	0.0721	0.0686	-	-	-	-
	teacher-guided	-	-	-	0.0793	-	0.0675	-	-	-
LSTM	standard	0.0588	0.0443	0.0324	-	-	-	0.0453	0.0293	0.062
	self-adaptive	-	0.0533	0.0377	-	-	-	0.0116	0.0483	0.062
	teacher-guided	-	0.0652	0.0451	-	-	-	0.0274	0.0705	0.0569

Table A.22: Perturbation matrix for the Soul dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	11.93	13.5	14.49	16.9	18.0	19.0	11.38	12.92	15.51
	self-adaptive	19.0	13.59	13.95	19.0	18.0	19.0	12.36	11.67	16.8
	teacher-guided	17.0	13.21	14.53	19.31	18.0	19.0	12.61	10.75	18.0
CNN	standard	-	-	20.5	16.51	-	-	-	-	-
	self-adaptive	-	-	-	19.0	18.0	-	-	-	-
	teacher-guided	-	-	-	22.0	-	19.0	-	-	-
LSTM	standard	15.0	12.67	8.57	-	-	-	12.18	8.0	17.0
	self-adaptive	-	15.5	10.17	-	-	-	3.0	12.92	17.0
	teacher-guided	-	19.67	12.0	-	-	-	7.16	19.0	15.51

Table A.23: Steps matrix for the Soul dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.9771	0.9918	0.9969	0.9853	1.0	0.971	0.9863	0.9778
	self-adaptive	0.0276	1.0	0.3738	0.3427	0.3382	0.3659	0.4174	0.3151	0.2741
	teacher-guided	0.0069	0.4008	1.0	0.3209	0.3382	0.3415	0.3683	0.2603	0.2296
CNN	standard	0.0	0.0047	0.0012	1.0	0.8333	0.8571	0.1818	0.0	0.0
	self-adaptive	0.0	0.0	0.0	0.0345	1.0	0.1429	0.0227	0.0	0.0
	teacher-guided	0.0	0.0	0.0	0.0172	0.1667	1.0	0.0	0.0	0.0
LSTM	standard	0.0082	0.0071	0.0047	0.1207	0.0833	0.2857	1.0	0.5	0.5455
	self-adaptive	0.0	0.0024	0.0024	0.0	0.0	0.1429	0.0682	1.0	0.3636
	teacher-guided	0.0	0.0012	0.0012	0.0	0.0	0.1429	0.1136	0.3333	1.0

Table A.24: ASR matrix for the Soul dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0863	0.0933	0.0891	0.1303	0.1279	0.1421	0.1114	0.0809	0.1016
	self-adaptive	0.1052	0.0936	0.0821	0.1317	0.1212	0.1391	0.1202	0.0991	0.1071
	teacher-guided	0.0953	0.0925	0.0892	0.1316	0.1192	0.1438	0.1219	0.0995	0.1077
CNN	standard	-	0.1188	0.076	0.13	0.1378	0.1452	0.1381	-	-
	self-adaptive	-	-	-	0.117	0.1315	0.1717	0.1056	-	-
	teacher-guided	-	-	-	0.0793	0.1348	0.1359	-	-	-
LSTM	standard	0.0958	0.0904	0.0791	0.1428	0.1148	0.1449	0.1149	0.0975	0.1141
	self-adaptive	-	0.0663	0.0567	-	-	0.1717	0.0913	0.0817	0.1113
	teacher-guided	-	0.0652	0.0672	-	-	0.1717	0.0983	0.0739	0.104

Table A.25: Perturbation matrix for the Soul dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	23.57	26.68	26.21	36.65	37.27	41.15	32.21	23.47	30.36
	self-adaptive	29.75	26.79	23.99	38.82	36.57	42.13	34.52	29.74	32.19
	teacher-guided	25.5	26.43	26.26	38.69	35.74	43.79	34.9	30.42	32.58
CNN	standard	-	32.33	20.5	36.56	40.91	42.03	38.62	-	-
	self-adaptive	-	-	-	33.0	38.74	50.0	28.0	-	-
	teacher-guided	-	-	-	22.0	38.56	39.27	-	-	-
LSTM	standard	26.0	25.53	23.57	40.47	35.0	42.5	33.22	27.38	35.4
	self-adaptive	-	19.0	18.12	-	-	50.0	24.77	23.67	33.46
	teacher-guided	-	19.67	21.33	-	-	50.0	26.57	23.32	31.35

Table A.26: Steps matrix for the Soul dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.9766	0.9883	1.0	1.0	-	0.9833	0.9737	0.976
	self-adaptive	0.0356	1.0	0.5987	0.2881	0.25	-	0.511	0.4737	0.497
	teacher-guided	0.0158	0.4971	1.0	0.322	0.25	-	0.4629	0.4298	0.491
CNN	standard	0.0	0.0119	0.0078	1.0	1.0	-	0.3368	0.1667	0.2
	self-adaptive	0.0	0.0024	0.0	0.1	1.0	-	0.1579	0.0833	0.1333
	teacher-guided	0.0	0.0012	0.0	0.1	0.0	-	0.0737	0.0833	0.0
LSTM	standard	0.0094	0.0119	0.01	0.2	0.0	-	1.0	0.6667	0.3333
	self-adaptive	0.0	0.0036	0.0033	0.1	0.0	-	0.3263	1.0	0.4667
	teacher-guided	0.0	0.0024	0.0022	0.1	0.0	-	0.2632	0.75	1.0

Table A.27: ASR matrix for the Soul dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.1455	0.153	0.1645	0.0855	0.0902	-	0.1498	0.1096	0.1216
	self-adaptive	0.2986	0.1526	0.1867	0.108	0.1657	-	0.1881	0.1865	0.1688
	teacher-guided	0.2333	0.1568	0.1645	0.1089	0.1657	-	0.194	0.198	0.1807
CNN	standard	-	0.1818	0.1446	0.0818	0.0902	-	0.22	0.3465	0.2517
	self-adaptive	-	0.1563	-	0.0605	0.0902	-	0.2321	0.3489	0.2667
	teacher-guided	-	0.1387	-	0.0605	-	-	0.2391	0.3489	-
LSTM	standard	0.199	0.1395	0.1521	0.096	-	-	0.164	0.1668	0.1549
	self-adaptive	-	0.1102	0.1829	0.1184	-	-	0.1958	0.1347	0.148
	teacher-guided	-	0.2286	0.2035	0.1184	-	-	0.1905	0.1236	0.1351

Table A.28: Perturbation matrix for the Soul dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.39	1.6	1.98	1.08	2.0	-	1.34	1.18	1.69
	self-adaptive	1.67	1.59	2.09	1.12	2.0	-	1.4	1.3	1.81
	teacher-guided	1.75	1.44	1.98	1.11	2.0	-	1.41	1.33	1.79
CNN	standard	-	1.69	2.09	1.11	2.0	-	1.55	2.0	3.0
	self-adaptive	-	1.57	-	1.0	2.0	-	1.31	2.0	3.0
	teacher-guided	-	1.33	-	1.0	-	-	1.19	2.0	-
LSTM	standard	1.0	1.47	2.05	1.0	-	-	1.37	1.36	1.47
	self-adaptive	-	1.33	2.91	1.0	-	-	1.51	1.25	1.9
	teacher-guided	-	2.0	3.38	1.0	-	-	1.49	1.22	1.85

Table A.29: Steps matrix for the Soul dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.9823	0.9927	1.0	1.0	1.0	0.9697	1.0	1.0
	self-adaptive	0.0124	1.0	0.4514	0.22	0.4167	0.5	0.3561	0.1579	0.2308
	teacher-guided	0.0062	0.4097	1.0	0.3	0.4167	0.25	0.2727	0.0789	0.2308
CNN	standard	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
	self-adaptive	0.0	0.0	0.0	0.1	1.0	0.0	0.0	0.0	0.0
	teacher-guided	0.0	0.0	0.0	0.1	0.0	1.0	0.0	0.0	0.0
LSTM	standard	0.0067	0.0053	0.0034	0.0	0.0	0.0	1.0	0.3333	0.3333
	self-adaptive	0.0	0.0036	0.0034	0.0	0.0	0.0	0.0833	1.0	0.3333
	teacher-guided	0.0	0.0018	0.0017	0.0	0.0	0.0	0.1667	0.3333	1.0

Table A.30: ASR matrix for the Soul dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0583	0.0597	0.0617	0.0674	0.0675	0.073	0.0587	0.0584	0.0657
	self-adaptive	0.0669	0.0599	0.0603	0.0721	0.0628	0.0723	0.0591	0.0568	0.0686
	teacher-guided	0.0696	0.0585	0.0617	0.0699	0.0628	0.0686	0.058	0.0515	0.0686
CNN	standard	-	-	-	0.0662	-	-	-	-	-
	self-adaptive	-	-	-	0.0738	0.0675	-	-	-	-
	teacher-guided	-	-	-	0.0819	-	0.073	-	-	-
LSTM	standard	0.0603	0.0571	0.0602	-	-	-	0.0605	0.0497	0.0641
	self-adaptive	-	0.0637	0.063	-	-	-	0.046	0.0584	0.0641
	teacher-guided	-	0.0712	0.0699	-	-	-	0.0489	0.0727	0.0657

Table A.31: Perturbation matrix for the Soul dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	15.22	17.23	17.72	23.9	25.0	28.0	14.97	12.26	22.21
	self-adaptive	27.0	17.47	16.14	28.27	25.0	28.0	16.17	10.0	26.89
	teacher-guided	26.0	16.08	17.82	27.0	25.0	28.0	16.11	1.0	26.89
CNN	standard	-	-	-	23.14	-	-	-	-	-
	self-adaptive	-	-	-	28.0	25.0	-	-	-	-
	teacher-guided	-	-	-	34.0	-	28.0	-	-	-
LSTM	standard	27.0	16.33	14.0	-	-	-	16.36	1.0	23.0
	self-adaptive	-	20.5	15.83	-	-	-	4.0	12.26	23.0
	teacher-guided	-	27.0	15.5	-	-	-	8.68	28.0	22.21

Table A.32: Steps matrix for the Soul dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.9787	0.9947	0.9929	1.0	1.0	0.9714	0.9844	0.9706
	self-adaptive	0.0315	1.0	0.4124	0.35	0.3636	0.2667	0.3857	0.2188	0.2157
	teacher-guided	0.0135	0.4323	1.0	0.3143	0.3333	0.2	0.319	0.1719	0.2157
CNN	standard	0.0	0.0054	0.0027	1.0	0.6667	0.6667	0.15	0.2	0.0
	self-adaptive	0.0	0.0	0.0	0.04	1.0	0.0	0.05	0.0	0.0
	teacher-guided	0.0	0.0	0.0	0.0	0.1667	1.0	0.0	0.0	0.0
LSTM	standard	0.0102	0.0068	0.0053	0.12	0.1667	0.0	1.0	0.4	0.5
	self-adaptive	0.0	0.0027	0.0027	0.04	0.0	0.0	0.1	1.0	0.375
	teacher-guided	0.0	0.0014	0.0013	0.04	0.0	0.0	0.15	0.4	1.0

Table A.33: ASR matrix for the Soul dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0989	0.1044	0.1056	0.1183	0.1113	0.107	0.1047	0.1094	0.1081
	self-adaptive	0.1087	0.1044	0.1037	0.1239	0.0983	0.117	0.1066	0.1149	0.1015
	teacher-guided	0.0926	0.1033	0.1055	0.1221	0.0997	0.1163	0.1091	0.1144	0.102
CNN	standard	-	0.113	0.106	0.1169	0.1174	0.1068	0.1179	0.1109	-
	self-adaptive	-	-	-	0.0983	0.1166	-	0.1187	-	-
	teacher-guided	-	-	-	-	0.1114	0.107	-	-	-
LSTM	standard	0.113	0.1112	0.1104	0.1376	0.1106	-	0.1054	0.1108	0.1041
	self-adaptive	-	0.1006	0.1124	0.1412	-	-	0.1041	0.1095	0.1103
	teacher-guided	-	0.1005	0.1335	0.1412	-	-	0.1	0.118	0.1081

Table A.34: Perturbation matrix for the Soul dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	19.92	23.32	22.43	36.96	38.24	38.0	24.27	25.19	29.27
	self-adaptive	35.29	23.48	19.08	41.02	38.17	35.75	26.01	35.43	28.59
	teacher-guided	27.67	21.83	22.49	39.7	37.0	32.0	26.85	35.09	31.77
CNN	standard	-	22.56	21.4	36.21	43.41	40.18	41.73	44.0	-
	self-adaptive	-	-	-	30.0	41.41	-	37.0	-	-
	teacher-guided	-	-	-	-	43.0	38.0	-	-	-
LSTM	standard	30.5	20.24	27.26	47.21	46.0	-	25.57	21.0	26.04
	self-adaptive	-	15.57	19.75	48.0	-	-	20.5	25.44	32.85
	teacher-guided	-	19.67	11.67	48.0	-	-	17.86	38.59	29.24

Table A.35: Steps matrix for the Soul dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	1.0	1.0	-	1.0	1.0	1.0	1.0
	self-adaptive	-	-	0.0	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	-	1.0	0.0	-	0.0	0.0	0.0	0.0
CNN	standard	-	-	0.0194	1.0	-	0.0	0.0	0.0	0.0
	self-adaptive	-	-	0.0097	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	-	0.0097	0.0	-	1.0	0.0	0.0	0.0
LSTM	standard	-	-	0.0291	0.1667	-	0.0	1.0	1.0	0.0
	self-adaptive	-	-	0.0097	0.0	-	0.0	0.0	1.0	0.0
	teacher-guided	-	-	0.0097	0.0	-	0.0	0.0	0.25	1.0

Table A.36: ASR matrix for the Spark dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	0.0534	0.0708	-	0.0756	0.0526	0.0695	0.0458
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	0.0534	-	-	-	-	-	-
CNN	standard	-	-	0.0534	0.0708	-	-	-	-	-
	self-adaptive	-	-	0.0534	-	-	-	-	-	-
	teacher-guided	-	-	0.0534	-	-	0.0756	-	-	-
LSTM	standard	-	-	0.0534	0.0795	-	-	0.0526	0.0695	-
	self-adaptive	-	-	0.0534	-	-	-	-	0.0695	-
	teacher-guided	-	-	0.0534	-	-	-	-	0.0784	0.0458

Table A.37: Perturbation matrix for the Spark dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	18.0	19.05	-	24.44	15.35	21.97	18.57
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	18.0	-	-	-	-	-	-
CNN	standard	-	-	18.0	19.05	-	-	-	-	-
	self-adaptive	-	-	18.0	-	-	-	-	-	-
	teacher-guided	-	-	18.0	-	-	24.44	-	-	-
LSTM	standard	-	-	18.0	21.0	-	-	15.35	21.97	-
	self-adaptive	-	-	18.0	-	-	-	-	21.97	-
	teacher-guided	-	-	18.0	-	-	-	-	25.0	18.57

Table A.38: Steps matrix for the Spark dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	1.0	1.0	-	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	0.0	0.0429	-	0.0405	0.0118	0.1603	0.2201
	teacher-guided	-	1.0	1.0	0.2302	-	0.5946	0.6003	0.0	0.444
CNN	standard	-	0.0777	0.0194	1.0	-	0.1316	0.0758	0.1429	0.1071
	self-adaptive	-	0.0097	0.0097	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	0.0291	0.0097	0.0	-	1.0	0.0	0.0	0.1429
LSTM	standard	-	0.3107	0.0291	0.5556	-	0.1053	1.0	0.7857	0.0714
	self-adaptive	-	0.1262	0.0097	0.0889	-	0.0526	0.0	1.0	0.1071
	teacher-guided	-	0.0583	0.0097	0.0	-	0.0789	0.0	0.0714	1.0

Table A.39: ASR matrix for the Spark dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0926	0.0534	0.1508	-	0.1199	0.1191	0.1096	0.095
	self-adaptive	-	0.0926	-	0.1692	-	0.1073	0.1373	0.1504	0.1178
	teacher-guided	-	0.0926	0.0534	0.1508	-	0.1336	0.1426	-	0.1149
CNN	standard	-	0.0926	0.0534	0.1508	-	0.1149	0.1406	0.1281	0.1178
	self-adaptive	-	0.0926	0.0534	-	-	-	-	-	-
	teacher-guided	-	0.0926	0.0534	-	-	0.1199	-	-	0.1137
LSTM	standard	-	0.0926	0.0534	0.1608	-	0.1178	0.1191	0.1077	0.08
	self-adaptive	-	0.0926	0.0534	0.1578	-	0.1277	-	0.1096	0.0959
	teacher-guided	-	0.0926	0.0534	-	-	0.1289	-	0.0784	0.095

Table A.40: Perturbation matrix for the Spark dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	44.0	18.0	42.39	-	41.56	34.86	35.42	38.44
	self-adaptive	-	44.0	-	48.84	-	46.07	47.12	49.48	48.07
	teacher-guided	-	44.0	18.0	46.25	-	44.94	41.66	-	46.14
CNN	standard	-	44.0	18.0	42.39	-	38.22	41.21	40.6	46.29
	self-adaptive	-	44.0	18.0	-	-	-	-	-	-
	teacher-guided	-	44.0	18.0	-	-	41.56	-	-	45.15
LSTM	standard	-	44.0	18.0	44.94	-	39.83	34.86	34.84	32.33
	self-adaptive	-	44.0	18.0	43.95	-	44.5	-	35.42	38.45
	teacher-guided	-	44.0	18.0	-	-	45.03	-	25.0	38.44

Table A.41: Steps matrix for the Spark dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	0.999	1.0	0.9988
	self-adaptive	-	1.0	-	0.622	0.7094	0.4672	0.6126	0.8857	0.8916
	teacher-guided	-	1.0	-	0.7063	0.819	0.6589	0.7141	0.9304	0.9236
CNN	standard	-	0.068	-	1.0	1.0	0.1375	0.4043	0.84	0.7284
	self-adaptive	-	0.0097	-	0.5446	1.0	0.025	0.2553	0.7	0.5926
	teacher-guided	-	0.0485	-	0.3861	0.8537	1.0	0.4043	0.82	0.8889
LSTM	standard	-	0.2816	-	0.7822	0.9634	0.3375	1.0	0.96	0.9136
	self-adaptive	-	0.1262	-	0.4257	0.6707	0.225	0.4787	1.0	0.8395
	teacher-guided	-	0.1359	-	0.198	0.5488	0.325	0.4149	0.86	1.0

Table A.42: ASR matrix for the Spark dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0692	-	0.2061	0.2177	0.1279	0.1636	0.2215	0.232
	self-adaptive	-	0.0692	-	0.2882	0.2777	0.225	0.2409	0.2458	0.2526
	teacher-guided	-	0.0692	-	0.2667	0.2564	0.1832	0.2165	0.2363	0.2464
CNN	standard	-	0.0693	-	0.2061	0.2177	0.1764	0.2743	0.2465	0.2593
	self-adaptive	-	0.0694	-	0.2373	0.2177	0.2166	0.3333	0.2645	0.2713
	teacher-guided	-	0.0693	-	0.2457	0.2237	0.1279	0.2706	0.2491	0.2476
LSTM	standard	-	0.0692	-	0.2209	0.2194	0.1611	0.1638	0.2251	0.2411
	self-adaptive	-	0.0693	-	0.2433	0.2345	0.1814	0.2524	0.2215	0.2449
	teacher-guided	-	0.0692	-	0.2338	0.2297	0.1596	0.2696	0.2425	0.232

Table A.43: Perturbation matrix for the Spark dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.03	1.05	2.04	1.06	1.0	1.0
	self-adaptive	-	1.0	-	1.01	1.05	2.18	1.06	1.0	1.0
	teacher-guided	-	1.0	-	1.01	1.06	2.12	1.06	1.0	1.0
CNN	standard	-	1.0	-	1.03	1.05	2.35	1.02	1.0	1.0
	self-adaptive	-	1.0	-	1.0	1.05	3.0	1.0	1.0	1.0
	teacher-guided	-	1.0	-	1.0	1.05	2.04	1.0	1.0	1.0
LSTM	standard	-	1.0	-	1.02	1.05	2.09	1.06	1.0	1.0
	self-adaptive	-	1.0	-	1.02	1.04	2.34	1.0	1.0	1.0
	teacher-guided	-	1.0	-	1.05	1.05	2.06	1.0	1.0	1.0

Table A.44: Steps matrix for the Spark dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	1.0	1.0	-	1.0	1.0	1.0	1.0
	self-adaptive	-	-	0.0	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	-	1.0	0.0227	-	0.0	0.0	0.0	0.0
CNN	standard	-	-	0.0194	1.0	-	0.0	0.0	0.0	0.0
	self-adaptive	-	-	0.0097	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	-	0.0097	0.0	-	1.0	0.0	0.0	0.0
LSTM	standard	-	-	0.0388	0.1667	-	0.0	1.0	1.0	0.0
	self-adaptive	-	-	0.0097	0.0	-	0.0	0.0	1.0	0.0
	teacher-guided	-	-	0.0097	0.0	-	0.0	0.0	0.25	1.0

Table A.45: ASR matrix for the Spark dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	0.0612	0.0749	-	0.0746	0.0594	0.0696	0.0454
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	0.0612	0.0765	-	-	-	-	-
CNN	standard	-	-	0.0597	0.0749	-	-	-	-	-
	self-adaptive	-	-	0.0542	-	-	-	-	-	-
	teacher-guided	-	-	0.0542	-	-	0.0746	-	-	-
LSTM	standard	-	-	0.0619	0.0827	-	-	0.0594	0.0696	-
	self-adaptive	-	-	0.0542	-	-	-	-	0.0696	-
	teacher-guided	-	-	0.0542	-	-	-	-	0.0765	0.0454

Table A.46: Perturbation matrix for the Spark dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	-	18.19	18.86	-	33.75	14.35	26.22	18.87
	self-adaptive	-	-	-	-	-	-	-	-	-
	teacher-guided	-	-	18.19	23.0	-	-	-	-	-
CNN	standard	-	-	17.38	18.86	-	-	-	-	-
	self-adaptive	-	-	20.0	-	-	-	-	-	-
	teacher-guided	-	-	20.0	-	-	33.75	-	-	-
LSTM	standard	-	-	19.18	22.0	-	-	14.35	26.22	-
	self-adaptive	-	-	20.0	-	-	-	-	26.22	-
	teacher-guided	-	-	20.0	-	-	-	-	39.0	18.87

Table A.47: Steps matrix for the Spark dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	1.0	1.0	-	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	0.0039	0.0356	-	0.03	0.0422	0.2545	0.2057
	teacher-guided	-	0.7236	1.0	0.2267	-	0.397	0.5156	0.0182	0.3714
CNN	standard	-	0.0196	0.0291	1.0	-	0.1429	0.0741	0.1667	0.0556
	self-adaptive	-	0.0098	0.0097	0.0	-	0.0	0.0	0.0	0.0
	teacher-guided	-	0.0098	0.0097	0.0	-	1.0	0.0	0.0	0.0556
LSTM	standard	-	0.1275	0.0583	0.4	-	0.1786	1.0	0.75	0.1667
	self-adaptive	-	0.049	0.0097	0.12	-	0.0714	0.0	1.0	0.1111
	teacher-guided	-	0.0294	0.0097	0.0	-	0.0357	0.0	0.0833	1.0

Table A.48: ASR matrix for the Spark dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.099	0.0891	0.1416	-	0.1172	0.121	0.1094	0.0914
	self-adaptive	-	0.099	0.094	0.1633	-	0.1488	0.1513	0.1328	0.1096
	teacher-guided	-	0.0963	0.0891	0.1401	-	0.1332	0.1333	0.0831	0.1041
CNN	standard	-	0.0939	0.0893	0.1416	-	0.1184	0.1292	0.1233	0.1059
	self-adaptive	-	0.0755	0.0676	-	-	-	-	-	-
	teacher-guided	-	0.0755	0.0676	-	-	0.1172	-	-	0.0938
LSTM	standard	-	0.0956	0.0867	0.1532	-	0.1107	0.121	0.1061	0.0879
	self-adaptive	-	0.0933	0.0676	0.1549	-	0.1138	-	0.1094	0.0883
	teacher-guided	-	0.091	0.0676	-	-	0.12	-	0.0969	0.0914

Table A.49: Perturbation matrix for the Spark dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	30.34	14.52	34.89	-	38.68	29.86	30.85	32.85
	self-adaptive	-	30.34	1.5	38.12	-	44.0	38.0	38.46	41.67
	teacher-guided	-	28.53	14.52	36.76	-	40.53	35.52	13.0	37.34
CNN	standard	-	25.2	15.64	34.89	-	38.97	35.18	36.2	40.0
	self-adaptive	-	22.5	7.5	-	-	-	-	-	-
	teacher-guided	-	22.5	7.5	-	-	38.68	-	-	45.0
LSTM	standard	-	29.89	13.54	39.78	-	35.45	29.86	28.22	33.31
	self-adaptive	-	28.92	7.5	43.17	-	37.44	-	30.85	33.89
	teacher-guided	-	28.89	7.5	-	-	45.0	-	21.0	32.85

Table A.50: Steps matrix for the Spark dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	-	1.0	1.0	0.9981	1.0	0.9532	0.9924
	teacher-guided	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
CNN	standard	-	0.4209	-	1.0	0.9143	0.9034	0.75	1.0	0.9647
	self-adaptive	-	0.4192	-	0.0326	1.0	0.5379	0.5714	0.9	0.5882
	teacher-guided	-	0.4198	-	0.0163	0.2952	1.0	0.5357	0.9	0.5294
LSTM	standard	-	0.4164	-	0.0272	0.0857	0.1103	1.0	0.625	0.2353
	self-adaptive	-	0.3493	-	0.0109	0.0952	0.0828	0.3214	1.0	0.1765
	teacher-guided	-	0.3823	-	0.0272	0.2952	0.2897	0.3571	0.7	1.0

Table A.51: ASR matrix for the Impala dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0294	-	0.059	0.0335	0.035	0.0429	0.035	0.0245
	self-adaptive	-	0.0294	-	0.059	0.0335	0.035	0.0429	0.0364	0.0246
	teacher-guided	-	0.0294	-	0.059	0.0335	0.035	0.0429	0.035	0.0245
CNN	standard	-	0.0107	-	0.0533	0.0337	0.0337	0.0431	0.036	0.0254
	self-adaptive	-	0.0107	-	0.034	0.0332	0.0295	0.0434	0.035	0.026
	teacher-guided	-	0.0107	-	0.0514	0.032	0.0347	0.0426	0.0376	0.0263
LSTM	standard	-	0.0106	-	0.0318	0.0296	0.0338	0.042	0.0369	0.0299
	self-adaptive	-	0.0106	-	0.0481	0.0277	0.0224	0.0412	0.036	0.0237
	teacher-guided	-	0.0106	-	0.037	0.0272	0.0258	0.0365	0.0349	0.0251

Table A.52: Perturbation matrix for the Impala dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	8.58	-	16.18	11.51	13.82	13.87	14.35	9.8
	self-adaptive	-	8.58	-	16.18	11.51	13.84	13.87	14.95	9.84
	teacher-guided	-	8.58	-	16.18	11.51	13.82	13.87	14.35	9.8
CNN	standard	-	3.12	-	15.13	11.71	13.51	14.74	14.86	10.43
	self-adaptive	-	3.11	-	12.6	11.44	12.41	15.16	14.52	10.89
	teacher-guided	-	3.12	-	18.67	10.75	13.73	14.93	15.49	10.97
LSTM	standard	-	3.1	-	12.03	9.47	11.04	14.06	15.39	13.21
	self-adaptive	-	3.08	-	24.5	10.03	10.2	15.18	14.86	10.44
	teacher-guided	-	3.1	-	16.62	10.32	12.09	13.66	14.5	10.27

Table A.53: Steps matrix for the Impala dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	-	1.0	1.0	0.9988	1.0	0.9767	0.9957
	teacher-guided	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
CNN	standard	-	0.4209	-	1.0	0.9429	0.867	0.8077	0.9867	0.953
	self-adaptive	-	0.4192	-	0.0108	1.0	0.4464	0.6154	0.88	0.6376
	teacher-guided	-	0.4198	-	0.0072	0.2743	1.0	0.5962	0.8133	0.5302
LSTM	standard	-	0.4164	-	0.0179	0.0971	0.103	1.0	0.5733	0.2953
	self-adaptive	-	0.3493	-	0.0036	0.0571	0.0558	0.2692	1.0	0.1544
	teacher-guided	-	0.3823	-	0.009	0.2057	0.2189	0.3269	0.7067	1.0

Table A.54: ASR matrix for the Impala dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0294	-	0.1158	0.0677	0.0611	0.0812	0.0611	0.052
	self-adaptive	-	0.0294	-	0.1158	0.0677	0.0611	0.0812	0.0625	0.0521
	teacher-guided	-	0.0294	-	0.1158	0.0677	0.0611	0.0812	0.0611	0.052
CNN	standard	-	0.0107	-	0.1071	0.0679	0.0576	0.0804	0.0594	0.0526
	self-adaptive	-	0.0107	-	0.034	0.0662	0.044	0.0803	0.0587	0.0579
	teacher-guided	-	0.0107	-	0.0663	0.0597	0.0601	0.0808	0.0583	0.0558
LSTM	standard	-	0.0106	-	0.0784	0.0771	0.0609	0.0777	0.0587	0.0679
	self-adaptive	-	0.0106	-	0.0481	0.0277	0.0293	0.0672	0.0598	0.0431
	teacher-guided	-	0.0106	-	0.037	0.0361	0.0346	0.0689	0.0586	0.053

Table A.55: Perturbation matrix for the Impala dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	8.58	-	32.94	22.97	23.78	27.63	27.15	22.44
	self-adaptive	-	8.58	-	32.94	22.97	23.81	27.63	27.75	22.52
	teacher-guided	-	8.58	-	32.94	22.97	23.78	27.63	27.15	22.44
CNN	standard	-	3.12	-	31.14	23.23	22.69	28.39	25.9	22.89
	self-adaptive	-	3.11	-	12.6	22.63	18.93	29.13	25.4	25.36
	teacher-guided	-	3.12	-	23.21	20.23	23.49	29.4	24.85	24.11
LSTM	standard	-	3.1	-	26.29	23.75	19.61	26.98	24.63	29.54
	self-adaptive	-	3.08	-	24.5	10.03	13.65	25.43	26.15	20.29
	teacher-guided	-	3.1	-	16.62	14.3	16.52	26.79	25.63	23.09

Table A.56: Steps matrix for the Impala dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	-	1.0	1.0	1.0	1.0	1.0	0.9977
	teacher-guided	-	1.0	-	1.0	1.0	1.0	1.0	1.0	0.5194
CNN	standard	-	0.3755	-	1.0	0.9655	0.6667	0.9421	1.0	0.1226
	self-adaptive	-	0.3736	-	0.0308	1.0	0.1667	0.7355	0.9444	0.0387
	teacher-guided	-	0.3742	-	0.0154	0.4483	1.0	0.7231	0.7963	0.0323
LSTM	standard	-	0.3718	-	0.0769	0.2069	0.0	1.0	0.0556	0.0452
	self-adaptive	-	0.3098	-	0.0154	0.0345	0.0	0.4504	1.0	0.0
	teacher-guided	-	0.3387	-	0.0462	0.069	0.0	0.0248	0.3519	1.0

Table A.57: ASR matrix for the Impala dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0198	-	0.0442	0.105	0.1093	0.324	0.1613	0.4106
	self-adaptive	-	0.0198	-	0.0442	0.105	0.1093	0.324	0.1613	0.4113
	teacher-guided	-	0.0198	-	0.0442	0.105	0.1093	0.324	0.1613	0.3696
CNN	standard	-	0.0064	-	0.0459	0.1052	0.116	0.324	0.1577	0.2294
	self-adaptive	-	0.0063	-	0.0345	0.1031	0.1945	0.3357	0.1598	0.1327
	teacher-guided	-	0.0064	-	0.0464	0.1103	0.1181	0.3383	0.1747	0.1711
LSTM	standard	-	0.0063	-	0.0324	0.1025	-	0.3187	0.0677	0.1405
	self-adaptive	-	0.0063	-	0.0464	0.1051	-	0.345	0.1577	-
	teacher-guided	-	0.0063	-	0.0248	0.0528	-	0.0496	0.1301	0.3948

Table A.58: Perturbation matrix for the Impala dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.09	1.33	1.91	1.05	1.1	6.13
	self-adaptive	-	1.0	-	1.09	1.33	1.91	1.05	1.1	6.12
	teacher-guided	-	1.0	-	1.09	1.33	1.91	1.05	1.1	6.14
CNN	standard	-	1.0	-	1.08	1.3	2.0	1.07	1.1	7.76
	self-adaptive	-	1.0	-	1.0	1.32	2.0	1.07	1.11	7.44
	teacher-guided	-	1.0	-	1.0	1.28	2.02	1.05	1.02	7.96
LSTM	standard	-	1.0	-	1.0	1.0	-	1.06	1.29	8.43
	self-adaptive	-	1.0	-	1.0	1.0	-	1.06	1.1	-
	teacher-guided	-	1.0	-	1.0	1.0	-	2.0	1.31	6.24

Table A.59: Steps matrix for the Impala dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	-	1.0	1.0	0.9991	1.0	1.0	0.9943
	teacher-guided	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
CNN	standard	-	0.4209	-	1.0	0.9245	0.8912	0.75	1.0	0.9765
	self-adaptive	-	0.4192	-	0.0435	1.0	0.517	0.5714	0.925	0.5882
	teacher-guided	-	0.4204	-	0.0163	0.3019	1.0	0.5357	0.9	0.5294
LSTM	standard	-	0.4164	-	0.0272	0.0849	0.1088	1.0	0.625	0.2353
	self-adaptive	-	0.3487	-	0.0109	0.0943	0.0816	0.3214	1.0	0.1647
	teacher-guided	-	0.3817	-	0.0272	0.283	0.2857	0.3571	0.75	1.0

Table A.60: ASR matrix for the Impala dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0354	-	0.0663	0.0461	0.0463	0.0519	0.0443	0.0383
	self-adaptive	-	0.0354	-	0.0663	0.0461	0.0463	0.0519	0.0443	0.0385
	teacher-guided	-	0.0354	-	0.0663	0.0461	0.0463	0.0519	0.0443	0.0383
CNN	standard	-	0.0291	-	0.0618	0.0458	0.0446	0.0494	0.0446	0.038
	self-adaptive	-	0.0291	-	0.0449	0.0459	0.0411	0.0487	0.0441	0.0374
	teacher-guided	-	0.0291	-	0.057	0.0451	0.0457	0.0465	0.0452	0.0374
LSTM	standard	-	0.0291	-	0.0467	0.0503	0.0511	0.0497	0.0446	0.037
	self-adaptive	-	0.0292	-	0.0509	0.0402	0.0356	0.0448	0.0446	0.0361
	teacher-guided	-	0.0291	-	0.0453	0.0419	0.0365	0.041	0.0438	0.0379

Table A.61: Perturbation matrix for the Impala dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	3.37	-	20.97	12.1	16.03	15.2	13.65	10.77
	self-adaptive	-	3.37	-	20.97	12.1	16.05	15.2	13.65	10.82
	teacher-guided	-	3.37	-	20.97	12.1	16.03	15.2	13.65	10.77
CNN	standard	-	1.18	-	18.58	12.27	15.0	15.47	13.7	10.7
	self-adaptive	-	1.18	-	10.64	12.0	13.7	15.83	13.54	10.39
	teacher-guided	-	1.18	-	20.0	10.65	15.66	14.86	14.06	10.38
LSTM	standard	-	1.17	-	7.96	10.95	12.62	15.02	13.53	10.9
	self-adaptive	-	1.17	-	17.0	9.44	10.9	15.45	13.7	10.24
	teacher-guided	-	1.17	-	13.0	11.25	12.34	13.01	13.63	10.63

Table A.62: Steps matrix for the Impala dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
	self-adaptive	-	1.0	-	0.9995	1.0	1.0	1.0	1.0	0.9966
	teacher-guided	-	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0
CNN	standard	-	0.4209	-	1.0	0.9554	0.8925	0.8222	0.9867	0.9514
	self-adaptive	-	0.4192	-	0.0231	1.0	0.4486	0.6667	0.9067	0.625
	teacher-guided	-	0.4209	-	0.0086	0.2739	1.0	0.6222	0.8	0.5347
LSTM	standard	-	0.4164	-	0.0231	0.0828	0.1121	1.0	0.5333	0.2778
	self-adaptive	-	0.3481	-	0.0058	0.0637	0.0701	0.3111	1.0	0.1597
	teacher-guided	-	0.3805	-	0.0144	0.2293	0.2336	0.3778	0.7333	1.0

Table A.63: ASR matrix for the Impala dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	0.0562	-	0.1059	0.0843	0.0792	0.0913	0.0813	0.0749
	self-adaptive	-	0.0562	-	0.1059	0.0843	0.0792	0.0913	0.0813	0.0751
	teacher-guided	-	0.0562	-	0.1059	0.0843	0.0792	0.0913	0.0813	0.0749
CNN	standard	-	0.0541	-	0.1019	0.0848	0.0782	0.089	0.0792	0.0742
	self-adaptive	-	0.0541	-	0.0715	0.0846	0.0724	0.0879	0.0794	0.0756
	teacher-guided	-	0.0541	-	0.091	0.0819	0.0793	0.0855	0.0774	0.0756
LSTM	standard	-	0.0541	-	0.0952	0.0932	0.0883	0.0877	0.077	0.0788
	self-adaptive	-	0.0543	-	0.0695	0.0607	0.0664	0.0787	0.0794	0.0687
	teacher-guided	-	0.0541	-	0.0671	0.0717	0.0658	0.0805	0.0776	0.0749

Table A.64: Perturbation matrix for the Impala dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	-	2.08	-	29.48	18.1	20.03	21.61	24.18	22.08
	self-adaptive	-	2.08	-	29.48	18.1	20.03	21.61	24.18	22.15
	teacher-guided	-	2.08	-	29.48	18.1	20.03	21.61	24.18	22.08
CNN	standard	-	1.11	-	26.75	18.57	19.4	23.06	21.94	21.35
	self-adaptive	-	1.11	-	8.3	18.2	16.86	23.97	21.84	22.42
	teacher-guided	-	1.11	-	26.24	15.55	19.97	23.51	19.77	21.79
LSTM	standard	-	1.11	-	22.58	16.29	16.01	21.17	18.65	24.17
	self-adaptive	-	1.11	-	10.0	6.43	13.31	21.02	22.23	18.37
	teacher-guided	-	1.11	-	9.57	13.12	14.17	23.17	21.78	21.94

Table A.65: Steps matrix for the Impala dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.5788	0.9101	0.9026	0.9327	0.9312	0.9194	0.9544	0.9776
	self-adaptive	0.5649	1.0	0.9127	0.9123	0.946	0.9825	0.9243	0.9658	0.9803
	teacher-guided	0.8621	0.8721	1.0	0.8352	0.7756	0.7492	0.6952	0.496	0.4973
CNN	standard	0.7369	0.7324	0.7808	1.0	0.7929	0.8363	0.8907	0.8832	0.9019
	self-adaptive	0.6945	0.6971	0.6867	0.2228	1.0	0.6847	0.7936	0.777	0.8084
	teacher-guided	0.6549	0.6722	0.5714	0.1432	0.2801	1.0	0.6953	0.6673	0.6986
LSTM	standard	0.6365	0.6349	0.5207	0.0979	0.2249	0.2853	1.0	0.446	0.4743
	self-adaptive	0.6789	0.6784	0.5903	0.1212	0.2406	0.3769	0.6314	1.0	0.6472
	teacher-guided	0.6549	0.6411	0.4286	0.0918	0.2249	0.3183	0.5921	0.5062	1.0

Table A.66: ASR matrix for the Forester dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0448	0.0522	0.0146	0.0424	0.0396	0.0388	0.0422	0.043	0.0357
	self-adaptive	0.0423	0.0472	0.0147	0.0424	0.0398	0.0391	0.042	0.0433	0.0359
	teacher-guided	0.0444	0.0473	0.017	0.042	0.0393	0.0399	0.0429	0.0427	0.0332
CNN	standard	0.0445	0.0495	0.0241	0.0423	0.0402	0.0381	0.0433	0.0422	0.0328
	self-adaptive	0.0446	0.0499	0.0251	0.0358	0.0394	0.0382	0.043	0.0427	0.0333
	teacher-guided	0.0448	0.05	0.0266	0.0382	0.0383	0.0382	0.0426	0.042	0.0326
LSTM	standard	0.044	0.0488	0.0275	0.0415	0.0383	0.0349	0.0429	0.0416	0.0339
	self-adaptive	0.0447	0.0496	0.0266	0.038	0.0377	0.0359	0.0421	0.0428	0.0334
	teacher-guided	0.0444	0.0498	0.0311	0.037	0.0363	0.0347	0.0429	0.0425	0.0332

Table A.67: Perturbation matrix for the Forester dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	12.27	13.65	4.25	13.11	13.19	13.67	13.14	13.09	12.7
	self-adaptive	11.82	13.03	4.27	13.15	13.23	13.86	13.09	13.17	12.73
	teacher-guided	12.25	13.08	4.97	13.07	12.99	13.62	13.19	13.78	12.35
CNN	standard	12.01	12.81	6.89	13.08	13.35	13.67	13.05	12.91	12.38
	self-adaptive	12.03	12.9	7.15	10.99	13.03	13.63	12.97	13.04	12.53
	teacher-guided	12.07	12.93	7.57	11.55	12.75	13.61	12.82	12.89	12.43
LSTM	standard	11.86	12.63	7.81	12.92	12.85	12.53	12.96	12.86	12.85
	self-adaptive	12.05	12.86	7.57	11.83	12.55	13.13	12.71	13.11	12.5
	teacher-guided	11.98	12.89	8.79	11.04	11.82	12.55	12.72	13.06	12.54

Table A.68: Steps matrix for the Forester dataset under BIM attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.5571	0.8778	0.8708	0.9109	0.9047	0.9205	0.9568	0.9806
	self-adaptive	0.423	1.0	0.8811	0.8799	0.9176	0.9388	0.9248	0.9627	0.9827
	teacher-guided	0.6474	0.8568	1.0	0.7797	0.7297	0.7112	0.6848	0.501	0.5086
CNN	standard	0.7981	0.7131	0.7672	1.0	0.8321	0.8018	0.9052	0.8841	0.9125
	self-adaptive	0.7684	0.6804	0.6754	0.1641	1.0	0.6213	0.8083	0.7601	0.8134
	teacher-guided	0.7409	0.646	0.5631	0.1059	0.2697	1.0	0.6992	0.6577	0.6976
LSTM	standard	0.7253	0.6048	0.5142	0.1031	0.2162	0.2561	1.0	0.4501	0.4659
	self-adaptive	0.7595	0.6478	0.5817	0.1248	0.2308	0.3333	0.6153	1.0	0.6306
	teacher-guided	0.7439	0.6186	0.4262	0.0912	0.2015	0.2695	0.5471	0.5211	1.0

Table A.69: ASR matrix for the Forester dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.0877	0.0775	0.0152	0.0728	0.0721	0.0669	0.0778	0.0774	0.0626
	self-adaptive	0.0665	0.0741	0.0153	0.0728	0.0715	0.0661	0.0776	0.0772	0.0626
	teacher-guided	0.0708	0.073	0.0209	0.0708	0.0703	0.068	0.0778	0.0769	0.0601
CNN	standard	0.0932	0.0645	0.0265	0.0735	0.0751	0.0643	0.0807	0.08	0.0601
	self-adaptive	0.0938	0.0631	0.0274	0.0523	0.073	0.062	0.0807	0.0798	0.0606
	teacher-guided	0.0947	0.0624	0.029	0.0561	0.0695	0.0665	0.0802	0.0796	0.0596
LSTM	standard	0.0951	0.062	0.03	0.0738	0.0702	0.0567	0.0796	0.0794	0.0592
	self-adaptive	0.0944	0.0627	0.0291	0.0703	0.0686	0.0574	0.0774	0.08	0.0593
	teacher-guided	0.0949	0.0632	0.0343	0.0705	0.0652	0.0549	0.0767	0.0813	0.0599

Table A.70: Perturbation matrix for the Forester dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	24.38	21.36	4.4	23.11	24.39	24.38	24.45	24.19	23.49
	self-adaptive	20.43	21.61	4.45	23.14	24.3	24.16	24.42	24.13	23.5
	teacher-guided	20.94	21.43	6.28	22.65	23.75	24.09	24.36	24.94	24.29
CNN	standard	24.3	17.33	7.65	23.46	25.33	23.65	24.57	24.98	24.09
	self-adaptive	24.42	16.88	7.85	16.68	24.6	22.65	24.51	24.89	24.28
	teacher-guided	24.65	16.63	8.26	17.62	23.69	24.35	24.22	24.87	24.12
LSTM	standard	24.75	16.57	8.56	24.23	24.25	21.63	24.25	25.03	23.2
	self-adaptive	24.58	16.76	8.29	23.2	23.41	21.77	23.46	25.05	23.26
	teacher-guided	24.72	16.92	9.74	22.4	22.19	20.38	23.0	25.56	23.96

Table A.71: Steps matrix for the Forester dataset under BIM attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.5749	0.2178	0.992	0.9313	0.8977	0.9141	0.9391	0.7818
	self-adaptive	0.2622	1.0	0.2514	0.9946	0.9702	0.9091	0.8524	0.9345	0.6818
	teacher-guided	0.4584	0.9098	1.0	0.8254	0.886	0.9062	0.6123	0.4982	0.5955
CNN	standard	0.6565	0.8395	0.7385	1.0	0.811	0.7568	0.9294	0.8977	0.8571
	self-adaptive	0.626	0.8196	0.7095	0.1892	1.0	0.6081	0.7738	0.7478	0.6429
	teacher-guided	0.5963	0.7977	0.6795	0.1227	0.378	1.0	0.5923	0.5879	0.5238
LSTM	standard	0.5925	0.774	0.6731	0.1164	0.2256	0.2838	1.0	0.4539	0.4762
	self-adaptive	0.6123	0.8091	0.701	0.1476	0.2378	0.4054	0.4385	1.0	0.619
	teacher-guided	0.604	0.7892	0.6945	0.1247	0.3049	0.3919	0.3239	0.3963	1.0

Table A.72: ASR matrix for the Forester dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.1817	0.2059	0.0379	0.0812	0.1026	0.0784	0.2698	0.1353	0.2352
	self-adaptive	0.0858	0.1545	0.0446	0.0811	0.1002	0.0702	0.2549	0.1317	0.2066
	teacher-guided	0.1348	0.159	0.0517	0.0743	0.0916	0.076	0.2554	0.1773	0.2355
CNN	standard	0.0952	0.1665	0.0368	0.0791	0.1025	0.0737	0.2345	0.1673	0.2121
	self-adaptive	0.0935	0.1678	0.036	0.0491	0.0978	0.0699	0.2274	0.1679	0.1969
	teacher-guided	0.0914	0.1695	0.0349	0.0663	0.0786	0.078	0.2104	0.17	0.2085
LSTM	standard	0.0936	0.1725	0.0353	0.0766	0.1028	0.0687	0.2331	0.1816	0.1872
	self-adaptive	0.0925	0.1695	0.0353	0.0686	0.1071	0.0646	0.1758	0.1632	0.1992
	teacher-guided	0.093	0.1725	0.035	0.0661	0.075	0.0602	0.1212	0.1578	0.2124

Table A.73: Perturbation matrix for the Forester dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.08	1.03	1.2	1.7	1.29	1.41	1.08	1.13	1.07
	self-adaptive	1.05	1.08	1.17	1.7	1.3	1.44	1.08	1.13	1.08
	teacher-guided	1.12	1.08	1.23	1.7	1.31	1.4	1.09	1.12	1.02
CNN	standard	1.05	1.01	1.1	1.64	1.3	1.62	1.17	1.12	1.09
	self-adaptive	1.06	1.01	1.08	1.34	1.3	1.5	1.17	1.12	1.13
	teacher-guided	1.06	1.01	1.06	1.33	1.27	1.51	1.2	1.11	1.09
LSTM	standard	1.05	1.01	1.05	1.41	1.3	1.38	1.17	1.1	1.17
	self-adaptive	1.06	1.01	1.07	1.34	1.26	1.75	1.25	1.12	1.13
	teacher-guided	1.05	1.01	1.06	1.36	1.3	1.55	1.34	1.09	1.08

Table A.74: Steps matrix for the Forester dataset under DeepFool attack with $\varepsilon = 0.01$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.482	0.9092	0.8603	0.8923	0.9004	0.9115	0.953	0.9738
	self-adaptive	0.6437	1.0	0.9123	0.9081	0.9326	0.9672	0.9226	0.9649	0.9803
	teacher-guided	0.8325	0.8592	1.0	0.8199	0.7498	0.7381	0.6952	0.5182	0.5913
CNN	standard	0.7383	0.7324	0.7839	1.0	0.8067	0.8385	0.8919	0.8848	0.9065
	self-adaptive	0.6931	0.6971	0.6859	0.2198	1.0	0.6862	0.7912	0.773	0.8084
	teacher-guided	0.6563	0.6722	0.5685	0.1416	0.2761	1.0	0.6929	0.6649	0.6963
LSTM	standard	0.6365	0.6328	0.5227	0.0965	0.2249	0.2892	1.0	0.4415	0.4836
	self-adaptive	0.6775	0.6784	0.5978	0.1221	0.2465	0.3815	0.6413	1.0	0.6472
	teacher-guided	0.6563	0.6411	0.4311	0.0916	0.2209	0.3169	0.5872	0.5018	1.0

Table A.75: ASR matrix for the Forester dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.057	0.063	0.0404	0.0537	0.0515	0.0501	0.0533	0.054	0.0472
	self-adaptive	0.0558	0.0585	0.0404	0.0531	0.0511	0.0499	0.0531	0.0541	0.0472
	teacher-guided	0.0565	0.0582	0.0415	0.0527	0.051	0.0507	0.0539	0.0528	0.0443
CNN	standard	0.0568	0.0604	0.0449	0.0531	0.0515	0.049	0.0542	0.0533	0.0441
	self-adaptive	0.0567	0.0605	0.0454	0.0499	0.0512	0.0491	0.0538	0.0536	0.0443
	teacher-guided	0.0566	0.0604	0.0462	0.0516	0.0498	0.0492	0.0539	0.0531	0.0438
LSTM	standard	0.0566	0.0598	0.0466	0.0522	0.0503	0.0468	0.054	0.0531	0.0443
	self-adaptive	0.0569	0.0602	0.0462	0.0502	0.0489	0.0473	0.053	0.0536	0.0448
	teacher-guided	0.0567	0.0605	0.0484	0.05	0.0494	0.0468	0.0543	0.0536	0.0442

Table A.76: Perturbation matrix for the Forester dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	14.42	17.24	5.14	13.94	15.95	17.13	14.74	13.76	16.53
	self-adaptive	13.21	15.65	5.15	13.53	15.55	16.97	14.6	13.87	16.55
	teacher-guided	14.19	15.7	6.0	13.45	15.21	16.35	14.9	15.29	16.44
CNN	standard	13.7	15.07	8.01	13.53	15.73	17.12	14.75	13.84	16.68
	self-adaptive	13.73	15.18	8.29	10.24	15.29	17.0	14.58	13.85	16.78
	teacher-guided	13.7	15.16	8.79	11.07	14.82	16.86	14.52	13.82	16.61
LSTM	standard	13.36	14.53	9.03	13.2	15.83	16.0	14.55	14.21	16.92
	self-adaptive	13.72	15.04	8.69	11.75	14.08	16.96	14.15	14.06	16.81
	teacher-guided	13.54	15.2	10.12	11.38	13.88	16.25	14.35	14.61	16.88

Table A.77: Steps matrix for the Forester dataset under PGD attack with $\varepsilon = 0.1$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	1.0	0.516	0.8903	0.82	0.8711	0.8619	0.913	0.9544	0.98
	self-adaptive	0.5071	1.0	0.8939	0.8486	0.8867	0.8836	0.9147	0.9572	0.9808
	teacher-guided	0.651	0.7912	1.0	0.7215	0.6937	0.6832	0.6676	0.5193	0.6414
CNN	standard	0.7953	0.7185	0.7746	1.0	0.8462	0.8332	0.9009	0.8818	0.9106
	self-adaptive	0.7641	0.6874	0.6763	0.1658	1.0	0.658	0.8017	0.738	0.7967
	teacher-guided	0.7355	0.6545	0.5565	0.1043	0.2636	1.0	0.6875	0.6361	0.6829
LSTM	standard	0.7178	0.6106	0.5234	0.1004	0.2119	0.2753	1.0	0.4577	0.5154
	self-adaptive	0.7515	0.66	0.6033	0.1237	0.2421	0.3431	0.632	1.0	0.6537
	teacher-guided	0.7372	0.6234	0.4293	0.0848	0.1967	0.2774	0.5385	0.4852	1.0

Table A.78: ASR matrix for the Forester dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	0.1058	0.1065	0.0742	0.0964	0.0918	0.0875	0.0948	0.0958	0.0807
	self-adaptive	0.0947	0.1001	0.0743	0.0951	0.0905	0.0864	0.0946	0.0955	0.0806
	teacher-guided	0.0975	0.0989	0.0762	0.0941	0.0909	0.0886	0.0955	0.0943	0.0778
CNN	standard	0.1118	0.1038	0.0807	0.095	0.0919	0.0857	0.0974	0.0969	0.078
	self-adaptive	0.1121	0.1036	0.0815	0.0866	0.0913	0.0855	0.0977	0.0967	0.0782
	teacher-guided	0.1124	0.1033	0.0828	0.0891	0.0899	0.0867	0.0976	0.0966	0.0775
LSTM	standard	0.1131	0.1032	0.0835	0.0938	0.0877	0.081	0.0973	0.0971	0.0791
	self-adaptive	0.1126	0.1035	0.0824	0.0906	0.0879	0.0802	0.0963	0.0968	0.0791
	teacher-guided	0.1125	0.1034	0.0861	0.0914	0.0868	0.0809	0.0972	0.0974	0.0779

Table A.79: Perturbation matrix for the Forester dataset under PGD attack with $\varepsilon = 0.2$.

		FCN			CNN			LSTM		
		standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided	standard	self-adaptive	teacher-guided
FCN	standard	22.14	19.68	7.71	19.9	22.44	23.31	20.34	20.62	22.84
	self-adaptive	17.0	19.76	7.72	19.38	21.93	22.96	20.22	20.57	22.82
	teacher-guided	18.76	19.48	8.92	19.12	21.71	22.33	20.29	20.8	23.79
CNN	standard	22.88	17.79	10.25	18.92	22.18	22.47	20.21	20.62	23.55
	self-adaptive	23.06	17.6	10.46	11.65	21.54	22.18	20.35	20.0	23.37
	teacher-guided	23.21	17.54	10.84	11.72	21.39	22.68	19.89	20.28	23.48
LSTM	standard	23.18	17.0	11.07	19.78	21.92	21.92	20.05	20.83	23.68
	self-adaptive	23.12	17.53	10.69	17.94	21.09	22.04	19.58	20.8	23.19
	teacher-guided	23.23	17.79	12.04	17.99	19.12	21.5	19.24	20.91	23.72

Table A.80: Steps matrix for the Forester dataset under PGD attack with $\varepsilon = 0.2$.

List of Figures

List of Tables