A Machine Learning Approach for Detecting GPS Location Spoofing Attacks in Autonomous Vehicles

S. Filippou*, A. Achilleos*, S. Z. Zukhraf*, C. Laoudias*, K. Malialis*, M. K. Michael*† and G. Ellinas*†
*KIOS Research and Innovation Center of Excellence, University of Cyprus, Nicosia, 1678, Cyprus
†Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, 1678, Cyprus

***Abstract*—Connected and Autonomous Vehicles (CAV) depend on satellite systems, such as the Global Positioning System (GPS), for location awareness. Location data are streamed in real-time to the CAV's perception engine from its onboard GPS receiver for autonomous driving and navigation. However, these receivers are vulnerable to location spoofing attacks that can be easily launched using Commercial-Off-The-Self (COTS) equipment and open-source software. Existing data-driven attack detection solutions typically require data associated with 'normal' and 'attack' labels. The latter are hard to collect in operational conditions or even in controlled experiments. To this end, we formulate the GPS location spoofing attack detection as an outlier detection problem. The proposed solution based on Machine Learning (ML) relies solely on normal location data for training during attack-free operation. Our solution demonstrates more than 98% detection accuracy according to standard metrics on realistic data produced with the CARLA driving simulator and outperforms by 15% another (non ML-based) state-of-the-art solution.**

***Index Terms*—Location spoofing, attack detection, machine learning, autonomous vehicles, cybersecurity.**

I. Introduction

Accurate location data are crucial for the safe operation of Connected and Autonomous Vehicles (CAV). These data are provided as input to the Advanced Driver Assistance System (ADAS) and the perception engine of the CAV enabling it to understand and interpret its surroundings; thus, supporting the autonomous driving and navigation functionalities, while preventing undesirable collisions with nearby vehicles and/or Vulnerable Road Users (VRU) like pedestrians, cyclists, etc. Location awareness is also important for the adoption of Vehicle-to-Vehicle/Infrastructure (V2V/V2I) protocols and the deployment of Vehicular Ad-hoc NETworks (VANET) [1], which are both important ingredients for wider acceptance of Intelligent Transportation Systems (ITS).

Reliable location information on the CAVs is of paramount importance for increased safety in ITS. Recently, however, threats against the security of autonomous driving have been recognized [2], raising concerns and questions about the trustworthiness of location data and how this could be ensured. Typically, CAVs rely on Global Satellite Navigation Systems (GNSS), such as the Global Positioning System (GPS), to determine their location through onboard GPS signal receivers. However, current GPS receivers are still susceptible to *jamming* and *spoofing* attacks. Both attacks are easy to implement and launch using Commercial-Off-The-Self (COTS) equipment, e.g., a laptop, a Software-Defined Radio board, an antenna, and a signal amplifier, and open-source software. Jamming attacks create disruptive signals that essentially block the satellite signals; while, they affect the availability of location data, such attacks are easy to detect [3]. On the other hand, location spoofing attacks deceive the user via the transmission of signals that have the same characteristics as those of the legitimate GPS satellite signals [4]. Such attacks are harder to detect and pose a serious threat to the safety of all actors in ITS.

Obviously, attacks first need to be detected to be able to defend against them. In order to detect GPS spoofing attacks, existing works either apply signal processing techniques [5], [6] that require additional hardware or follow data-driven approaches [7]–[10] that depend on the availability of GPS data. In many cases, however, GPS data need to include ground truth labels, i.e., normal or attack. Labeled data are hard to acquire in real-life scenarios, especially attack data from previous known attacks that are extremely rare or hard to reproduce in controlled scenarios for experimentation. To this end, this work introduces a data-driven approach for detecting GPS location spoofing attacks based on Machine Learning (ML). Specifically, the contribution of this work is threefold:

- We formulate the GPS location spoofing attack against autonomous vehicles, for the first time, as an anomaly detection problem. In this way, only attack-free training data are required to train any underlying ML algorithm that is able to learn the normal behavior and identify any deviation as anomaly (i.e., attack).
- We investigate thoroughly different design options and assess various ML algorithms for the anomaly detector in a simulation environment implemented in the real-time CARLA simulator [11] for autonomous driving systems.
- We test and evaluate the proposed solution in terms of the G-mean and F1-score metrics through multiple scenarios using realistic data. Our solution demonstrates more than 98% attack detection accuracy, and outperforms by 15% another (non ML-based) state-of-the-art solution.

This work was supported by the European Union's H2020 research and innovation programme under the CARAMEL project (Grant agreement No 833611). It has also been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

The rest of the paper is structured as follows. Section II overviews the related works on detecting GPS location attacks. The system model is described in Section III, while Section IV presents the framework for ML-based attack detection. Comparative results of various ML-based outlier detection methods are reported and discussed in Section V. Finally, Section VI provides concluding remarks and directions for future work.

II. Related Works

In general, solutions for detecting GPS location spoofing attacks can be categorized as *signal processing* and *data-driven* solutions. For instance, in [5], real-time attack detection is implemented using Software-Defined Radios (SDR) by analyzing the phase difference of the GPS signals collected by different antennas. In [6], the vehicles use dedicated short-range communication to exchange the GPS pseudo-range measurements with other vehicles in the vicinity. Each vehicle uses these measurements to compute statistics that enable the local detection of possibly spoofed GPS signals on the high correlation of their arrival times. These locally detected signals are forwarded to a head vehicle that uses the minimum-maximum change detection procedure to optimize global spoofing detection. The main limitation of signal processing solutions is the requirement for additional hardware.

On the other hand, data-driven attack detection solutions process data that are readily available by COTS GPS receivers and try to learn the attack behavior. Such data include raw GPS data such as number of satellites, pseudo-range measurements, Signal-to-Noise Ratio (SNR) values, etc., or the computed GPS location coordinates. For instance, authors in [7] compare the vehicle's accelerometer readings against the estimated acceleration measurements of the GPS device, with the mismatch beyond a certain threshold signifying a GPS spoofing attack. In [8], Multi-Sensor Fusion (MSF) algorithms are employed to estimate the vehicle's location by partially relying on the GPS readings. Leveraging measurements from neighboring vehicles, collaborative mechanisms are able to detect GPS location spoofing attacks within Vehicular Ad-hoc NETworks (VANET) by means of MSF [9]. In the collaborative detection paradigm, the measurements include the relative distances, the relative angles, and the relative azimuth angles among the vehicles, as well as the absolute position measurements (i.e., GPS positions) of all vehicles. In our previous works [10] we employed in-vehicle multi-sensory data (e.g., accelerometer, gyroscope, steering angle, etc.) to estimate the GPS-free locations of the vehicle and compared them with the corresponding locations reported by the vehicle's GPS receiver. If the deviation exceeds a pre-defined threshold (selected during attack-free operation), then an attack is detected.

With the increasing volume of sensory data produced by vehicles, ML methods have also been applied for detecting location spoofing attacks. Surprisingly, there exists limited work in the area of autonomous vehicles; thus, we overview existing works from related domains (e.g., autonomous aerial vehicles). Typically, ML methods are grouped in supervised methods that require ground truth information, i.e., data labeled as either *normal* or *attack* in our case, and unsupervised methods that do not rely on such information.

Regarding supervised ML methods, authors in [12] utilize k-Nearest Neighbor, and Support Vector Machine (SVM) for the detection and classification of location spoofing misbehavior based on features such as location and movement plausibility check. In [13], an artificial Neural Network (NN) is used to detect GPS spoofing signals on Unmanned Aerial Vehicles (UAVs). Information from the incoming GPS signals are collected to create the input features for the NN, including the number of satellites, SNR, and doppler shift. In [14], the authors compared various ML methods for detecting jamming attacks in wireless networks. Specifically, they investigated different signal features to recognize jamming signals and applied the Random Forest, SVM, and NN models. Authors in [15] provide an extensive review of *misbehavior detection* methods in cooperative ITS focusing on jamming and fake message injection attacks. In [16], a method for detecting spoofing of the GPS signals is presented that employs a multi-layer NN. Even though these methods are quite effective, the requirement for labeled data, especially attack data, limits their applicability in practice.

To this end, some works have explored unsupervised ML methods and have approached the intended task through anomaly detection. In [17], an autoencoder, which is a special type of NN, has been proposed for fault detection on UAVs. Five categories of features are used, including internal measurements, location, position, orientation, system status, and control. Further, an unsupervised multivariate Gaussian-based anomaly detection algorithm is employed in [18] to spot unusual driving behaviors on semi-autonomous vehicles, based on accelerometer and GPS sensors of manually driven automobiles. Finally, in [19], the authors proposed the use of an autoencoder followed by a one-class SVM to secure software-defined networks from malicious traffic, by training the models using only samples from the normal classes. To the best of our knowledge, we formulate for the first time the problem of GPS location spoofing in autonomous vehicles as an anomaly detection problem.

III. System Model

The main actor in our sytem model for the GPS location spoofing scenario is a CAV moving on the road network having location awareness through its onboard GPS receiver. The receiver computes the location of the CAV using satellite signals. There is also a wireless network infrastructure (e.g., cellular towers of telecommunication operators, Wi-Fi Access Points or routers, etc.) that can be used for the provision of GPS-free location information. Finally, there is an attacker that attempts to spoof the location of the vehicle, e.g., causing it to divert to an undesirable trajectory with potentially harmful consequences, as illustrated in Fig. 1. In this context, the location information that is readily available on the CAV (both from the GPS receiver and the surrounding wireless network infrastructure) and the location attack are modeled as follows.

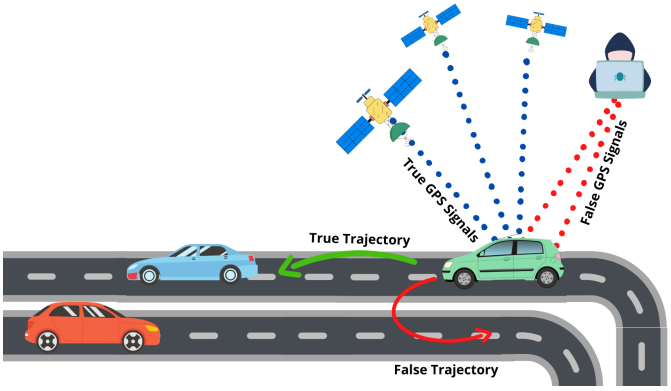

Fig. 1: GPS location spoofing attack scenario on CAVs.

A. Location Model

Assume a CAV that moves on the road network, as shown in Fig. 1, and its true location and velocity are, respectively, $\mathbf{p}_k = [x_k, y_k]^\top$ and $\mathbf{u}_k = [\dot{x}_k, \dot{y}_k]^\top$, where x_k and y_k are the location coordinates at time instance k. The CAV is equipped with a GPS receiver that processes the satellite positioning signals and outputs the GPS location of the CAV $\mathbf{p}_k^G = [x_k^G, y_k^G]^\top$, which is modeled as a Gaussian random variable $\mathbf{p}_k^G \sim \mathcal{N}(\mathbf{p}_k, \mathbf{\Sigma}_k^G)$, where $\mathbf{\Sigma}_k^G = \text{diag}_2(\sigma_k^G)$ is the 2×2 diagonal covariance matrix of the GPS readings with standard deviation of noise $\sigma_k^G = \sigma^G$ in both coordinates.

The CAV is also equipped with a device for the collection of location-dependent network measurements from the surrounding wireless network infrastructure and connected vehicles, independently from the GPS measurements, e.g., based on SDR technology for scanning specific frequency bands. Such devices are typical in CAVs for enabling V2V/V2I communications. The device implements a Localization Algorithm (LA) that estimates the current location of the CAV based on these signals, i.e., using network-assisted LAs based on radio signal measurements, e.g., timing, angle, or signal strength measurements, from the neighboring transmitters (e.g, cellular towers, Wi-Fi access points, etc.) or using the information received from connected vehicles by applying cooperative LA; see [20] for an overview of such algorithms. For instance, the LA could leverage ambient Signals-of-Opportunity (SoO) to compute the CAV location, as described in [21].

The device outputs the estimated location of the CAV $\mathbf{p}_k^L = [x_k^L, y_k^L]^\top$ modeled as a Gaussian random variable $\mathbf{p}_k^L \sim \mathcal{N}(\mathbf{p}_k, \mathbf{\Sigma}_k^L)$, where $\mathbf{\Sigma}_k^L = \text{diag}_2(\sigma_k^L)$ is the covariance matrix that denotes the uncertainty of the LA, i.e., the noise of the CAV's locations estimated by the LA has standard deviation $\sigma_k^L = \sigma^L$ in both coordinates. Note that the estimated locations $\mathbf{p}_k^L$ can be used directly as GPS-free raw location measurements. Alternatively, locations $\mathbf{p}_k^L$ can be refined by means of Bayesian filtering, e.g., Extended Kalman Filter (EKF) combined with an underlying mobility model, e.g., bicycle model, that fuses them with in-vehicle sensory data (e.g., accelerometer, gyroscope, steering angle, etc.) to improve location accuracy as reported in [10]. One might argue that if an attacker goes all the way to spoof GPS signals, then he/she could also attack $\mathbf{p}_k^L$. However, attacking $\mathbf{p}_k^L$ would be much more difficult, e.g., due to the protection and authentication mechanisms in cellular networks or difficulty is attacking multiple signals when SoO are used.

B. Attack Model

Assume an attacker who uses COTS equipment, including Software Defined Radio (SDR) hardware, amplifier, and antenna, combined with open-source SDR software, to interfere with the legitimate GPS signals and disturb the original GPS data. The COTS equipment can be either mounted on an Unmanned Aerial Vehicle (UAV) or user-carried at the ground level, as shown in Fig. 1. Similarly to [10], it is assumed that the attacker spoofs the GPS location and introduces a user-defined constant bias value in both GPS location coordinates.

Following the definition of the attack-free GPS location above, the GPS location of the CAV under attack is modeled as a Gaussian random variable $\mathbf{p}_k^G \sim \mathcal{N}(\mathbf{p}_k + \mathbf{B}_A, \mathbf{\Sigma}_k^G)$, where $\mathbf{B}_A = b[1,1]^\top$ is the attack vector and b denotes the attack bias, i.e., the magnitude of the attack to each coordinate (in terms of meters). Thus, $b = 0$ represents the attack-free case. Note that during the attack, and while the CAV moves inside the attack range, the GPS receiver reports the spoofed location (i.e., $b \neq 0$), which the CAV perceives as its valid location, if it is not equipped with a reliable spoofing detection solution.

IV. Machine Learning-Based Attack Detection

A. Problem Formulation

In this work, we formulate the problem of GPS location spoofing in autonomous vehicles as an anomaly detection problem. Let the location from the GPS readings at time step k be $\mathbf{p}_k^G = [x_k^G, y_k^G]$, and the estimated CAV location be $\mathbf{p}_k^L = [x_k^L, y_k^L]$. A feature generating process creates the differential feature $\mathbf{d_k} \in \mathbf{R}^2$, corresponding to the positional vector defined as

$$\mathbf{d_k} = \mathbf{p}_k^G - \mathbf{p}_k^L = [x_k^d, y_k^d], \tag{1}$$

where $x_k^d = x_k^G - x_k^L$ and $y_k^d = y_k^G - y_k^L$.

We consider a learning model (anomaly detector) $f : \mathbf{R}^2 \rightarrow \{0, 1\}$, which receives an input $\mathbf{d_k} \in \mathbf{R}^2$ at time step k and predicts its label $L_k \in \{0, 1\}$, such that $L_k = f(\mathbf{d_k})$; the value $L_k = 0$ corresponds to an attack-free location, while $L_k = 1$ corresponds to a spoofed location.

In order to take into consideration the temporal aspects of the data, for spoofing attacks presented in/present at previous time steps, a sliding window is introduced to help with the prediction task, i.e., $L_k = f(\mathbf{d}_k, \mathbf{d}_{k-1}, \ldots, \mathbf{d}_{k-W})$, where W is the sliding window size.

B. Attack Detection Pipeline

The pipeline of the proposed approach for detecting GPS location spoofing attacks is depicted in Fig. 2. In the *Training* stage the anomaly detector f is trained to learn the normal behavior of the data. In order to achieve this we utilize the

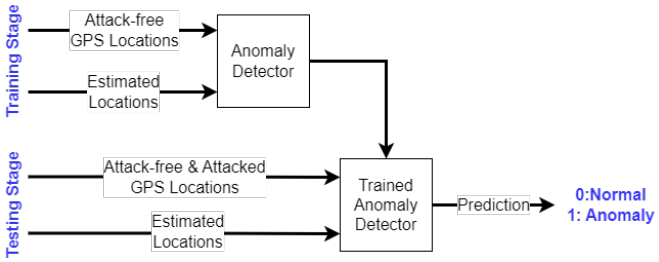

Fig. 2: Machine learning-based anomaly detection pipeline.

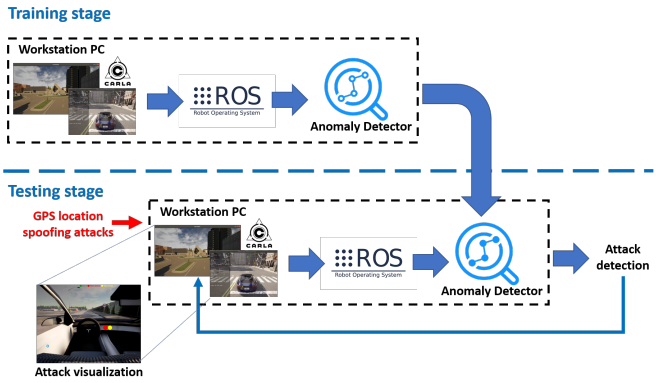

Fig. 3: Experimental setup.

train dataset D_{tr}, which consists only of attack-free locations, that is, $D_{tr} = \{\mathbf{d}_k\}_{k=1}^{|D_{tr}|}$.

Part of the training process is to determine which learning model and its hyper-parameters are more suitable. For this, we consider the validation set D_{val}, which is considerably smaller than the size of the training set, i.e., $|D_{val}| << |D_{tr}|$. This separate dataset, i.e., $D_{val} = \{(\mathbf{d}_k, L_k)\}_{k=1}^{|D_{val}|}$, consists of attack-free locations and a much smaller number of attacked locations.

The *Testing* stage is performed on an independent test dataset, i.e., data which the model did not encounter during the training and validation stages. It consists of attack-free and attacked locations, defined as $D_t = \{(\mathbf{d}_k, L_k)\}_{k=1}^{|D_t|}$.

V. Performance Evaluation

A. Experimental Setup

Our experimental setup is based on a Linux-based workstation PC with 8 GB RAM and GPU running the CARLA simulator which is a development, training, and validation tool for autonomous driving systems [11]. At the *Training* stage a moving vehicle is instantiated in the simulation environment driving under normal conditions (i.e., no attack) along a predefined trajectory with user-selected noise profiles for the simulated GPS data and the GPS-free estimated vehicle locations (e.g., based on cellular networks). The simulated sensor data are collected and forwarded, via a publish-subscribe protocol implemented using the Robot Operating System (ROS), to the (untrained) Anomaly Detector developed in Python for training the underlying ML model, as depicted in Fig. 3.

In the *Testing* stage, a moving vehicle is simulated driving along different trajectories under normal and attack scenarios assuming noise profiles for the simulated GPS data and the GPS-free estimated vehicle locations similar to the *Training* stage. The attack prediction labels L_k are sent back to the CARLA simulator for verification. To this end, a simple visualization interface was implemented within CARLA as two lights inside the vehicle's cockpit (see bottom left in Fig. 3). The right light provides the ground truth with respect to the attack status, i.e., it is turned off under normal conditions and turns yellow when an attack is actually ongoing. The left light reflects the attack detection, i.e., it is green when no attack is detected and turns red when attacks are detected. While we are able to collect the simulation data including the attack prediction labels to analyze offline, this interface provides an easy way to assess the performance of various attack detection solutions in real-time.

B. Simulation Parameters & Datasets

Different test cases are investigated for which the attack bias b in the GPS measurements varies for different trajectories traversed by the vehicle. The other parameters are the same for all test cases including $\sigma^L = 10\,\text{m}$, $\sigma^G = 3\,\text{m}$ (i.e., the standard deviation for locations $\mathbf{p}_k^L$ and $\mathbf{p}_k^G$ provided by the network-based LA and the GPS, respectively), and sampling interval $\Delta t = 0.05\,\text{s}$. In summary, we produced seven trajectories T_i, $i = 1, \ldots, 7$, as shown in TABLE I, corresponding to ten datasets, as T_6 and T_7 were simulated with various values for the attack bias. All trajectories are depicted in Figure 4.

The data are split into train, validation, and test sets. The train set includes three normal datasets that represent T_2, T_3 and T_4 without any "spoofed" data. Recall that since we have formulated the problem as an anomaly detection problem, a learning model is trained using *only* normal data, i.e., without any GPS spoofing attack. The validation set consists of two "spoofed" datasets that represent T_1 and T_5 with attack bias $b = 5$. The test set consists of five "spoofed" datasets that represent T_6 and T_7 with attack biases $b = \{5, 6, 9\}$, and $b = \{5, 9\}$, respectively. Notice that the attack biases encountered in the test set are different from the bias ($b = 5$) each model experienced in training (indirectly through the validation set).

No	Total	Normal	Attacked	Attack bias [m]
T_1	6,020	3,302	2,718	5
T_2	13,433	13,433	0	0
T_3	10,265	10,265	0	0
T_4	11,730	11,730	0	0
T_5	4,542	2,272	2,270	5
T_6	4,193	2,097	2,096	5, 6, 9
T_7	9,129	4,566	4,563	5, 9

TABLE I: Number of total, normal, and attacked data points in each trajectory.

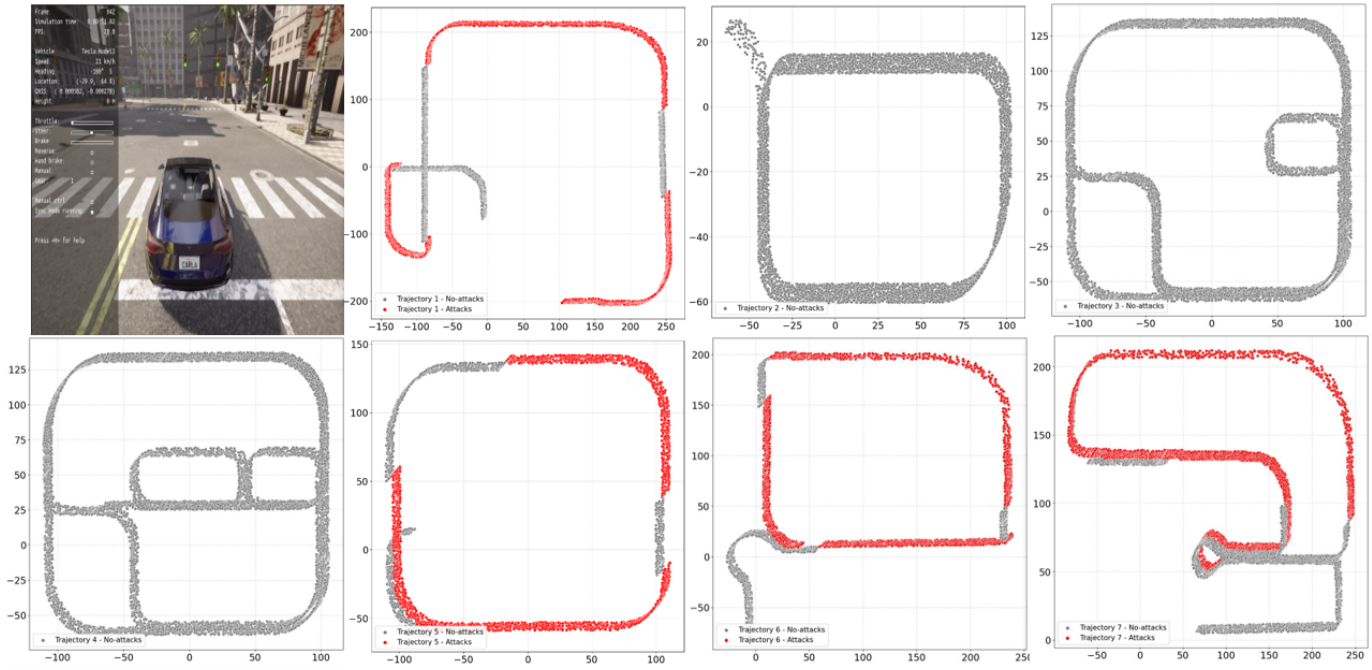

Fig. 4: The CARLA simulation environment (top left) and the seven trajectories used in our experiment, i.e., T_1, T_5, T_6, and T_7 with attack bias $b = 5$ and T_2, T_3, and T_4 with attack bias $b = 0$.

C. Performance Metrics

We analyze the detection results of the proposed solution using a **confusion matrix**. The confusion matrix classifies the results into *True Positive* (T_P), *True Negative* (T_N), *False Positive* (F_P), and *False Negative* (F_N). We assess the performance of our attack detection solution in terms of two popular metrics, namely the harmonic mean (or F1-score) and the geometric mean (G-mean). These are both appropriate metrics as they are less sensitive to class imbalance [22]; recall that our problem suffers from severe imbalance, as anomalies (i.e., attacks) constitute rare events.

The **F1-score (F1)** is defined as the harmonic mean of Precision and Recall. Specifically, Precision (P) provides information concerning the rate at which the algorithm detects attacks. In our application scenario, it is the ratio of the true attacks detected over all detected attacks including false positives given by

$$P = \frac{\#\ of\ true\ attacks\ detected}{\#\ of\ true\ and\ false\ attacks\ detected} = \frac{T_P}{T_P + F_P}. \tag{2}$$

Similarly, Recall (R) is the ratio of the true attacks detected over all true attacks including the missed detections, i.e., false negatives. This is given by

$$R = \frac{\#\ of\ true\ attacks\ detected}{\#\ of\ true\ attacks} = \frac{T_P}{T_P + F_N}. \tag{3}$$

Finally, the F1-score ($F1$) is the weighted average of P and R that measures accuracy on the data set given by

$$F1 = 2\left(\frac{P \times R}{P + R}\right). \tag{4}$$

$F1$ gets a higher value (near 1) when the F_P and F_N are low. If a system is performing poorly by generating more F_P and F_N, the $F1$-score will be low (near 0).

The **G-mean** (G_m) is defined as the geometric mean of Recall and Specificity and is given by

$$G_m = \sqrt{R \times S}, \tag{5}$$

where the Specificity (S), which is defined as the true negative rate, is given by

$$S = \frac{T_N}{T_N + F_P}. \tag{6}$$

Note that G_m has the desirable property of being high when both R and S are high, and when their difference is small [22].

D. Compared Methods

Our experimental study considers various ML-based anomaly detection methods as well as a traditional data-driven (non-learning) state-of-the-art method. All methods that we include in our comparison are described below.

Isolation Forest (iForest) [23]: An anomaly detection algorithm that isolates anomalies by building an ensemble of trees, for which anomalies are identified as those which have short average path lengths on trees.

Local Outlier Factor (LOF) [24]: An outlier detection algorithm that calculates for a given sample the local density deviation with respect to its neighbors. An anomaly corresponds to a sample that has a lower density than its neighbors.

One Class Support Vector Machine (OC-SVM) [25]: An unsupervised algorithm that separates data using a hyper-sphere boundary, which is created from the available data. An anomaly is any data point that lies outside that boundary.

AutoEncoder (AE) [26]: A special type of neural network that is often used for anomaly detection. An AE consists of an encoder followed by a decoder and attempts to copy its input. Specifically, the reconstructed error of normal data is minimized after training. An anomaly is detected if the reconstructed error is larger than a pre-specified threshold.

Threshold-based Attack Detection (TAD) [10]: A data-driven (non-learning) method that during the threshold selection phase uses the GPS locations $\mathbf{p}_k^G$ and the estimated CAV locations $\mathbf{p}_k^L$ to compute their deviation by means of Euclidean distance assuming attack-free conditions. A threshold is selected based on the percentage of F_P that are tolerable under normal conditions. During standard operation, an attack is signified if the average deviation between the GPS locations and the estimated CAV locations within a given time window exceeds the pre-selected threshold.

To facilitate the reproducibility of the results, in this work TABLE II provides the values of the hyper-parameters for each method, after tuning on the validation set. For the implementation of iForest, LOF, and OC-SVM algorithms we have used the popular scikit-learn library. The reader is directed towards the library's documentation [27] for more information on each algorithm's hyper-parameters and their default values. The TAD method is implemented in Python according to [10].

Algorithm	**Hyper-parameters**
iForest	Num. of Estimators = 100, Maximum num. of features = 0.9, Maximum num. of samples = 0.8, Proportion of outliers = 0.04
LOF	Num. of neighbors = 1000, Leaf size = 1, Proportion of outliers = 0.02
OC-SVM	Kernel = Radial Basis Function, NU: Upper bound of training errors and Lower bound of support vectors = 0.01, Kernel coefficient γ = 0.1
AE	Num. of layers = 3, Num. of neurons = (256,128,32), Learning rate = 0.0001, Mini-batch size = 64, Num. of epochs = 200
TAD	Percentage of F_P for selecting the threshold $1-\gamma = 5\%$, window size $w = 5$

TABLE II: Hyper-parameters used for each method.

E. Simulation Results

In this subsection, we present and discuss the simulation results. These can be reproduced using the software capsule that is released openly on the Code Ocean reproducible research platform[1].

1) Role of the sliding window: In this section we examine the impact of the sliding window size on the performance of the tested ML-based anomaly detection methods. TABLE III presents the G_m and $F1$ obtained on the validation set for all the methods given window sizes equal to 0, 3, 5, 10, 20.

[1]https://doi.org/10.24433/CO.0550935.v1

As we can see from TABLE III, performance is very low when we do not take into consideration any previous data (i.e., $\mathbf{w} = \mathbf{0}$), compared to varying window sizes. As the window size increases, performance increases up to a point, beyond which it declines again. This happens because the short-term temporal correlations among sequential data points help avoid erroneous oscillations between 'normal' and 'attack' detection; however, considering many past data points does not work well when the ground truth switches from 'normal' to 'attack' and vice versa. The best window sizes for iForest, LOF, OC-SVM, and AE are 5, 10, 10, and 20, respectively. Results of the iForest and AE models are averaged over 50 repetitions to address the stochastic nature of the algorithms.

Window size	Metric/ Algorithm	iForest	LOF	OC-SVM	AE
w=0	G_m	87.41	78.75	85.29	86.97
	$F1$	86.69	76.56	84.22	86.13
w=3	G_m	97.54	96.03	96.95	91.93
	$F1$	97.56	95.96	96.91	91.61
w=5	G_m	**97.78**	97.69	97.19	95.05
	$F1$	**97.80**	97.68	97.16	94.88
w=10	G_m	97.60	**97.85**	**97.36**	94.34
	$F1$	97.63	**97.84**	**97.33**	93.96
w=20	G_m	97.30	97.71	97.15	**95.49**
	$F1$	97.34	97.72	97.14	**95.31**

TABLE III: G-mean and F1-score results obtained by each algorithm for different window sizes on the validation set.

2) Role of the learning model: In this section, we investigate the performance of the ML-based outlier detection methods, using a window size equal to 10, as this window size provides the best performance for most of our methods. In particular, the LOF model, achieved the highest performance with $G_m = 97.85\%$ and $F1 = 97.84\%$, while the AE algorithm achieved the lowest performance with a $G_m = 94.34\%$ and $F1 = 93.96\%$. The performance of OC-SVM and iForest was slightly worse than the LOF model.

3) Comparative study: For the comparison between the ML-based methods and the data-driven TAD method for detecting location spoofing attacks, we selected the best performing ML-based method, i.e., LOF. As mentioned earlier, we have used a window size of 10 and therefore we removed the first 10 data from TAD in order to have the same amount of data when testing our algorithms. As we can see in TABLE IV, LOF significantly outperforms TAD in terms of all performance metrics. Notice that the values in TABLE IV correspond to the performance on the test set, unlike the ones in TABLE III which correspond to the validation set. Specifically, attack detection using the LOF outlier detection method achieved $G_m = 98.43\%$ and $F1 = 98.45\%$, i.e., around 15% better than TAD which achieved $G_m = 83.49\%$ and $F1 = 83.92\%$ on the test set. Table V presents the confusion matrix of the LOF and TAD methods on the test set. We observe that the proposed solution based on the LOF model, reduces false detections and misdetections significantly, i.e., F_P decreases approximately $7\times$ and even more importantly F_N is reduced approximately $20\times$.

Method	P	R	S	$\mathbf{G_m}$	F1
LOF	97.61	99.31	97.56	**98.43**	**98.45**
TAD	82.08	85.85	81.20	**83.49**	**83.92**

TABLE IV: Precision, Recall, Specificity, G-mean, and f1-score results on the test set.

Method	$\mathbf{T_N}$	$\mathbf{F_P}$	$\mathbf{F_N}$	$\mathbf{T_P}$
LOF	14,998	375	107	15,307
TAD	12,483	2,890	2,181	13,233

TABLE V: Confusion matrix of LOF adn TAD on the test set.

VI. Conclusion

We treat GPS location spoofing attacks as anomalies and develop an ML-based anomaly detection solution for identifying such attacks. The proposed solution does not require any attack data during the training stage and achieves remarkable classification accuracy in terms of $F1$ and G_m, ranging from 95% to 98% in both metrics depending on the underlying ML model. The variant that relies on the LOF model achieved the highest detection accuracy amongst all ML-based techniques investigated, as well as a 15% higher detection accuracy as compared to a non ML-based state-of-the-art solution.

For our future work we plan to implement the proposed solution on an embedded computing device (e.g., Nvidia Jetson) and deploy it on one of our CAVs at scale that is part of our research infrastructure at the KIOS Center of Excellence, University of Cyprus[2]. This will give us the opportunity to test the solution while the vehicle is moving and a GPS location spoofing attack is launched using COTS SDR hardware and open-source spoofing software. We also plan to investigate *online* ML-based algorithms that are able to adapt on-the-fly to changing conditions, e.g., lower uncertainty in GPS locations in open-sky rural areas compared to urban areas, without degrading detection accuracy or increasing false positives.

References

[1] R. K. Jaiswal and C. Jaidhar, "A performance evaluation of location prediction position-based routing using real GPS traces for VANET," *Wireless Personal Communications*, vol. 102, no. 1, pp. 275–292, 2018.

[2] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 357–372, 2019.

[3] R. Ferreira, J. Gaspar, P. Sebastião, and N. Souto, "Effective GPS jamming techniques for UAVs using low-cost SDR platforms," *Wireless Personal Communications*, vol. 115, no. 4, pp. 2705–2727, 2020.

[4] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[5] J. Friedt, W. Feng, D. Rabus, and G. Goavec-Merou, "Real time GNSS spoofing detection and cancellation on embedded systems using software defined radio," in *Proc. 15th European Conference on Antennas and Propagation (EuCAP)*, 2021.

[6] F. A. Milaat and H. Liu, "Decentralized detection of GPS spoofing in vehicular ad hoc networks," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1256–1259, 2018.

[7] K.-C. Kwon and D.-S. Shim, "Performance analysis of direct GPS spoofing detection method with AHRS/Accelerometer," *Sensors*, vol. 20, no. 4, pp. 1–22, 2020.

[8] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with Devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing," in *Proc. 29th USENIX Security Symposium*, 2020, pp. 931–948.

[9] M. Kamal, C. Kyrkou, N. Piperigkos, A. Papandreou, A. Kloukiniotis, J. Casademont, N. P. Mateu, D. B. Castillo, R. D. Rodriguez, N. G. Durante, P. Hofmann, P. Kapsalas, A. S. Lalos, K. Moustakas, C. Laoudias, T. Theocharides, and G. Ellinas, "A comprehensive solution for securing connected and autonomous vehicles," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 790–795.

[10] M. Kamal, A. Barua, C. Vitale, C. Laoudias, and G. Ellinas, "GPS location spoofing attack detection for enhancing the security of autonomous vehicles," in *Proc. IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021, pp. 1–7.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conference on Robot Learning (CoRL)*, 2017, pp. 1–16.

[12] S. So, P. Sharma, and J. Petit, "Integrating plausibility checks and machine learning for misbehavior detection in VANET," in *Proc. 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 564–571.

[13] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of GPS spoofing attacks on unmanned aerial systems," in *Proc. 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019, pp. 1–6.

[14] Y. Arjoune, F. Salahdine, M. S. Islam, E. Ghribi, and N. Kaabouch, "A novel jamming attacks detection approach based on machine learning for wireless communication," in *Proc. IEEE International Conference on Information Networking (ICOIN)*, 2020, pp. 459–464.

[15] R. W. van der Heijden, S. Dietzel, T. Leinmüller, and F. Kargl, "Survey on misbehavior detection in cooperative intelligent transportation systems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 779–811, 2019.

[16] E. Shafiee, M. R. Mosavi, and M. Moazedi, "Detection of spoofing attack using machine learning based on multi-layer neural network in single-frequency GPS receivers," *The Journal of Navigation*, vol. 71, no. 1, pp. 169–188, 2018.

[17] K. H. Park, E. Park, and H. K. Kim, "Unsupervised fault detection on unmanned aerial vehicles: Encoding and thresholding approach," *Sensors*, vol. 21, no. 6, pp. 1–17, 2021.

[18] C. Ryan, F. Murphy, and M. Mullins, "Semiautonomous vehicle risk analysis: A telematics-based anomaly detection approach," *Risk analysis*, vol. 39, no. 5, pp. 1125–1140, 2019.

[19] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using LSTM based autoencoder," in *Proc. 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020, pp. 37–45.

[20] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A survey of enabling technologies for network localization, tracking, and navigation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3607–3644, 2018.

[21] N. Souli, P. Kolios, and G. Ellinas, "Online relative positioning of autonomous vehicles using signals of opportunity," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 4, pp. 873–885, 2022.

[22] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[24] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, 2000.

[25] D. Tax and R. Duin, "Support vector data description," *Machine Learning*, vol. 54, pp. 45–66, 2004.

[26] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. ACM Workshop on Machine Learning for Sensory Data Analysis*, 2014, p. 4–11.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 2011, pp. 2825–2830, 2011.

[2]https://www.kios.ucy.ac.cy/intelligent-transportation-systems/