IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 25 September 2023; revised 29 December 2023 and 31 March 2024; accepted 2 April 2024.
Date of publication 10 April 2024; date of current version 16 April 2024.

The associate editor coordinating the review of this article and approving it for publication was A. Eltawil.

Digital Object Identifier 10.1109/TMLCN.2024.3386649

A Deep Learning Based Induced GNSS Spoof Detection Framework

ASIF IQBAL[1] (Member, IEEE), MUHAMMAD NAVEED AMAN[2] (Senior Member, IEEE), AND BIPLAB SIKDAR[1] (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583
[2]School of Computing, University of Nebraska-Lincoln, Lincoln, NE 68523 USA

CORRESPONDING AUTHOR: M. N. AMAN (naveed.aman@unl.edu)

This work was supported in part by the Ministry of Education, Singapore, under Grant T2EP20121-0036 (Tier 2).

ABSTRACT The Global Navigation Satellite System (GNSS) plays a crucial role in critical infrastructure by delivering precise timing and positional data. Nonetheless, the civilian segment of the GNSS remains susceptible to various spoofing attacks, necessitating robust detection mechanisms. The ability to deter such attacks significantly enhances the reliability and security of systems utilizing GNSS technology. Supervised Machine Learning (ML) techniques have shown promise in spoof detection. However, their effectiveness hinges on training data encompassing all possible attack scenarios, rendering them vulnerable to novel attack vectors. To address this limitation, we explore representation learning-based methods. These methods can be trained with a single data class and subsequently applied to classify test samples as either belonging to the training class or not. In this context, we introduce a GNSS spoof detection model comprising a Variational AutoEncoder (VAE) and a Generative Adversarial Network (GAN). The composite model is designed to efficiently learn the class distribution of the training data. The features used for training are extracted from the radio frequency and tracking modules of a standard GNSS receiver. To train our model, we leverage the Texas Spoofing Test Battery (TEXBAT) datasets. Our trained model yields three distinct detectors capable of effectively identifying spoofed signals. The detection performance across simpler to intermediate datasets for these detectors reaches approximately 99%, demonstrating their robustness. In the case of subtle attack scenario represented by DS-7, our approach achieves an approximate detection rate of 95%. In contrast, under supervised learning, the best detection score for DS-7 remains limited to 44.1%.

INDEX TERMS Generative adversarial network, GNSS security, GPS spoofing, receiver security, representation learning, spoof detection, TEXBAT, variational autoencoder.

I. INTRODUCTION

GLOBAL Navigation Satellite System (GNSS) has become deeply ingrained in our daily lives, permeating various sectors of national economies and societal functions. Its extensive civilian applications span smart grids, smart cities, financial systems, UAV navigation, communications, safety-critical operations, and precision agriculture, while it also plays a pivotal role in military domains, including aerospace, maritime operations, and precision-guided weaponry [1]. The widespread adoption of GNSS can be attributed to the open access of civilian GNSS signals, which has enabled the development of cost-effective GNSS receivers, readily accessible to the general public [2], [3]. However, the very characteristics that make GNSS appealing, i.e., low power and openness, also render it vulnerable to Radio Frequency Interference (RFI). This interference can manifest as either intentional or unintentional disruptions, with the former category further divided into jamming and spoofing interference [4]. Among these, spoofing interference represents a particularly deceptive threat to GNSS integrity. Malicious actors can exploit spoofing techniques to deceive GNSS receivers, leading them to report inaccurate timing and positional data [4]. Given the importance of GNSS in numerous critical applications, ensuring its security and reliability has emerged as a paramount concern, sparking extensive research endeavors within the scientific community [1]. Integration of GNSS receivers across the aforementioned diverse applications amplifies the impact

of GNSS spoofing on their performance. Various spoofing detection methods have been proposed in these domains, for instance, [5], [6], [7] for smart grids, [8], [9] for UAVs, and [10], [11] for terrestrial vehicles. Our research focuses on detecting spoofing attacks directly at the GNSS receiver level, contrasting with techniques that conduct spoof detection at the application side. From this point forward, we will exclusively focus on spoofing detection methods that conduct detection at the receiver level. By addressing spoofing at the source, downstream applications reliant on GNSS data can operate without susceptibility to such attacks.

Detection methods for GNSS spoofing fall into three primary categories: (i) cryptographic approaches that rely on the unpredictability and verifiability of signal modulation within GNSS spreading codes or navigation data, employing cryptographic principles for signal authenticity verification; (ii) geometric techniques that leverage the angle-of-arrival diversity exhibited by genuine GNSS signals, using spatial characteristics for distinguishing authentic from spoofed signals; and (iii) GNSS signal processing techniques that include methods that do not fit into the previous two categories, often involving signal processing techniques for spoof detection [12]. Among these techniques, single antenna based techniques offer practical advantages, such as requiring no modifications to GNSS signals, low-cost implementation, and software/firmware updates at the receiver end. These approaches include direct power monitoring [13], utilization of Automatic Gain Control (AGC) units [14], and monitoring the auto-correlation profile of a receiver's tracking loop through Signal Quality Monitoring (SQM) [15], [16], [17], [18], [19], [20]. Certain methods combine power monitoring and auto-correlation analysis [12], [21], with the majority adopting a Bayesian detection framework for spoof detection. These diverse approaches constitute a growing research field dedicated to safeguarding GNSS against spoofing attacks. In this paper, we contribute to this area by introducing a GNSS spoof detection model based on representation learning.

Detecting GNSS spoofing attacks poses a significant challenge, primarily attributed to the varying power advantages exhibited by the spoofed signals over genuine ones. In cases where spoofed signals enjoy a substantial power advantage, conventional SQM metrics tend to remain relatively stable. However, an abnormally high received power or Carrier-to-Noise Spectral Density Ratio (C/N_0) can raise immediate suspicion regarding the presence of a spoofed signal at the receiver. Conversely, when the power advantage is minimal, the attack may either prove ineffective or lead to significant distortion of the auto-correlation function. This distortion arises from the interaction between authentic and spoofed signals with similar power levels. In such scenarios, SQM metrics can effectively detect these attacks during their initial stages, particularly when the correlation peak carry-off begins, as demonstrated in Section II-C. However, it's essential to recognize that SQM metrics are best suited for transient detection and may lose their effectiveness once the spoofed signal displaces the peak by approximately one chip duration of the respective satellite's pseudo random number (PRN) code. Similarly, received power and C/N_0 behave as static indicators. It is a reasonable assumption that a spoofer cannot entirely obstruct or negate the authentic GNSS signal. Therefore, a robust defense strategy against spoofing attacks emerges by combining abnormal power detection with SQM metrics within a unified detection scheme. This approach has been previously explored in works such as [12] and [21].

Prior studies in single antenna based spoof detection have demonstrated effectiveness against spoofing attacks involving medium to low power advantages relative to the genuine signal [9], [12], [21]. However, addressing attack scenarios where the spoofer achieves a full carrier phase alignment with the genuine signal during both initial insertion and tracking correlator peak carry-off stage poses a significant challenge. These attacks demand either physical access to the victim receiver or precise channel information between the attacker's and victim's antennas [22], [23]. Termed as 'sophisticated,' such attacks are the most difficult to execute. Yet, given the widespread use of civilian GNSS signals in critical infrastructure, attackers seeking substantial gains may invest in equipment to execute such sophisticated attacks. Single-antenna-based GNSS receivers face considerable difficulty in detecting these attacks. In this study, we introduce a detection model capable of identifying both intermediate and sophisticated attacks. Leveraging genuine data samples, our proposed model is trained within a representation learning framework, enabling effective discernment of whether a test sample originates from the same distribution as the training data. Our comprehensive analysis across various attack scenarios demonstrates the exceptional detection performance of our proposed model across all attack scenarios.

A. RELATED WORKS

The Bayesian frameworks utilized in the aforementioned studies demand careful selection of signal models, prior and/or likelihood functions, and rely on the detection of feature distribution changes induced by the presence or absence of a spoofed signal. Only then, can a suitable detection threshold be confidently determined to achieve a desired level of detection or false alarm probabilities. In contrast, Machine Learning (ML) methods, specifically deep learning (DL) models like Fully Connected Neural Networks (FCNNs) [24], have showcased their proficiency in tackling complex challenges that defy conventional modeling techniques. These approaches reduce the requirements of precise specifications of mathematical models, noise models, or their associated statistical parameters. It is precisely this attribute that positions ML and DL techniques as well-suited for addressing the challenge of spoofing detection.

In recent years, there has been a notable surge in the development of ML-based techniques for GNSS spoofing detection. These approaches utilize features extracted

from standard GNSS receivers' acquisition, tracking, and Position-Velocity-Time (PVT) modules as input for training various supervised ML models in classification tasks. For instance, Bose et al. [25] employed an FCNN that incorporated features such as C/N_0, pseudorange, carrier phase, and Doppler shift to identify spoofing attempts. Similarly, Aissou et al. [9] used a set of 13 features extracted from tracking and PVT solution blocks to train multiple ML classifiers. In another work, Shafiee et al. [26] explored the use of FCNN, Naive Bayes, and K-Nearest Neighbors (K-NN) algorithms, incorporating received power alongside two SQM metrics (delta and early-late phase) for classifier training. Manesh et al. [27] employed an FCNN-based classifier with inputs consisting of pseudorange, Doppler shift, and Signal-to-Noise Ratio (SNR). Similarly, Iqbal et al. [28] used received power, C/N_0, and five SQM metrics to train an ensemble of ML classifiers for detecting time-push attacks in smart grid systems. Borhani et al. [29] leveraged the Cross Ambiguity Function (CAF) acquired from the acquisition block of a GNSS receiver to train deep learning models, including an FCNN, for detecting multiple peaks in the CAF, indicating the presence of a spoofed signal. A similar approach was presented by Li et al. [30] using a Generative Adversarial Network (GAN). These methods, however, are computationally expensive due to the requirement for multiple CAF computations. Dasgupta et al. [31] used the PVT solution from the GPS receiver, speed and steering angle data from a Control Area Network (CAN), and directional acceleration values from the intertial measurement unit (IMU) as input features to train an LSTM model. Subsequently, they used the feature forecasting to flag whether the current feature values are closer to the predictions or not. Similarly, Calvo et al. [10] used the Doppler shift of the GPS satellites to train an LSTM for spoof detection. Finally, Kim et al. [11] derived differential features from PVT module-calculated location data to identify irregularities in mobility profiles. They trained multiple ML-based classifiers, achieving a noteworthy detection accuracy of up to 99.1%. A summary of ML and DL based spoof detection methods is given in Table 1.

However, it's worth noting that some works, such as [9], [11], [25], and [31], incorporate PVT solutions as features. However, this inclusion can introduce additional detection delays since PVT solutions are acquired over a look-back sample window. For instance, even under hot start conditions, where the receiver possesses valid time, position, almanac, and ephemeris data, it can still take an upwards of 22 seconds to re-acquire a complete PVT fix [32], [33], with the subsequent solutions generated over a few seconds. However, by the time these computations are completed, a spoofer may have already gained full control over the tracking stage. Moreover the PVT solution may not yield additional information if the spoofer ensures that the induced changes in time and coordinate solutions remain within the expected mobility profile of the target receiver.

While ML-based methods (discussed previously) have demonstrated their effectiveness in countering GNSS spoofing attacks, they exhibit certain limitations:

- These methods face a fundamental challenge inherent in supervised learning: the acquisition and labeling of training datasets that comprehensively cover the various attack classes. This necessity to encompass as many attack classes as possible within the training datasets poses a significant challenge.
- Another notable challenge arises when these models encounter datasets that substantially deviate from or contain data entirely unseen during their training phase. This limitation often manifests when the models have been exposed to only a limited range of spoofing scenarios, lacking diversity in their training data. As a result, they may struggle to effectively recognize and adapt to new and more complex spoofing scenarios.
- Effective feature selection is crucial, impacting not only the model's detection capabilities but also its speed and accuracy, emphasizing the need for features that can be quickly acquired and are highly informative.
- Rather than utilizing established spoofing benchmarks like the TEXBAT dataset [22], [23] or similar resources, many of these methods rely on proprietary datasets generated internally. This makes the performance comparison across works rather difficult.

B. OUR CONTRIBUTIONS

To address the aforementioned limitations, in this work:

- We develop a detection model trained within a representation learning framework, diverging from supervised learning methods. This strategy resolves the primary challenge associated with supervised learning by focusing solely on acquiring high-quality genuine data for training, rather than attempting to encompass the entire attack space.
- Our detection model is proficient in identifying both intermediate and sophisticated level attacks, broadening its capability beyond basic threats.
- Our model is trained using an informative featureset easily obtainable from any standard GNSS receiver during run-time.
- We perform an evaluation conducted on the publicly available TEXBAT dataset, encompassing a variety of attack patterns, from simplistic to sophisticated spoofing scenarios, facilitating straightforward comparisons with existing works.

In this paper, we employ a feature vector comprising received power, C/N_0, and five distinct SQM metrics at each time point. As depicted in Figure 1, received power is calculated at the RF block output, while the remaining features are readily accessible (or computable) at the tracking block of a standard GNSS receiver, ensuring practicality and cost-effectiveness. By using static and transient detection features,

TABLE 1. Summary of related work on GNSS spoof detection using ML and DL based models.

Paper	Input Features	ML Models	Receiver Type		Attacks	Data Type	Sophisticated Attack
			Static	Dynamic			
Shafiee *et al.* [26]	Received power, delta, and early-late phase metrics	FCNN, K-NN, Naïve Bayes	✓	✗	1	Inhouse	✗
Manesh *et al.* [27]	Pseudorange, Doppler shift, SNR	FCNN	✓	✗	1	Inhouse	✗
Borhani *et al.* [29]	CAF from Acquisition block	FCNN and CNN	✓	✗	1	Synthetic	✗
Li *et al.* [30]	CAF from Acquisition block	GAN and CNN	✓	✗	1	Synthetic	✗
Aissou *et al.* [9]	13 features with PVT solution	K-NN, RN, SVMs	✓	✗	3	Inhouse	✗
Bose *et al.* [25]	C/N_0, pseudorange, carrier phase, Doppler shift	FCNN	✓	✗	1	Inhouse	✗
Dasgupta *et al.* [31]	PVT from GPS receiver, speed, steering angle from CAN, and acceleration from IMU	LSTM	✗	✓	1	Comma-2k19	✗
Calvo *et al.* [10]	Doppler shifts	LSTM	✗	✓	1	Inhouse	✗
Kim *et al.* [11]	PVT solution & its derivatives	K-NN, SVM, RFC, XGB, FCNN, and AE	✗	✓	4	VeReMi	✗
Iqbal *et al.* [28]	Received power, C/N_0, 5 SQM metrics	RFC, SVM, K-NN, FCNN, XGB	✓	✗	3	TEXBAT	✓
Proposed	Received power, C/N_0, 5 SQM metrics	RL based Model	✓	✓	6	TEXBAT	✓

Static or Dynamic GNSS receiver, Control Area Network (CAN), Inertial Measurement Unit (IMU), Vehicular Reference Misbehavior (VeReMi).
Fully Connected Neural Network (FCNN), K-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Random Forest Classifier (RFC),
Convolutional Neural Network (CNN), Generative Adversarial Network (GAN), Auto-Encoder (AE), Radius Neighbours (RN),
Extreme Gradient Boost (XGB), Long Short-Term Memory (LSTM), Representation Learning (RL).
Inhouse: Retransmission and reception of modified received GPS signals. Synthetic Data: Simulated genuine and spoofed GPS datasets.
Attacks: The number of different attack scenarios studied.

our featureset, empowers the detection of spoofed signals both before and after the correlation peak carry-off phase.

Our primary focus is on evaluating the detection performance and resilience of ML classifiers against various spoofing attacks. For performance baseline, we use 4 supervised ML models: the Random Forest Classifier (RFC), Gradient Boost (GB), K-Nearest Neighbours (K-NN), the Support Vector Machine (SVM), and a DL based FCNN. These models are analysed under classic supervised learning and under leave-one-out testing scenario, highlighting their limitations. The central contribution of this work lies in the development of a representation learning-based zero-day spoof detection model. Our proposed model consists of three distinct FCNNs, connected as composite model containing a Variational AutoEncoder (VAE) and a Generative Adversarial Network (GAN). The model is trained using a unified algorithm that combines both reconstruction and adversarial training frameworks. Once fully trained, the model offers three distinct detection strategies, each surpassing baseline performance. The most effective detection model achieves an impressive 99% detection accuracy for simple to intermediate-level spoofing scenarios and maintains a detection accuracy of over 93% for the sophisticated attack scenario, all while imposing low computational demands.

To summarize, the major contributions of this paper are as follows:

- We comprehensively evaluate the performance and resilience of multiple ML models, including RFC, GB, K-NN, SVM, and an FCNN under different training and spoofing conditions. This analysis provides insights into the limitations of these classifiers when dealing with evolving spoofing threats.
- Our primary contribution is the development of a modular representation learning-based zero-day spoof detection model which is formulated by combining a Variational AutoEncoder (VAE) and a Wasserstein Generative Adversarial Network (WGAN).
- Our fully trained model offers three distinct detection strategies with different computational requirements, each outperforming the baseline classifiers. These strategies enhance the reliability and accuracy of GNSS spoofing detection.
- We conduct a comprehensive analysis by evaluating our detection strategies across various spoofing scenarios, encompassing time and position shifts. These scenarios involve both static and mobile GNSS receivers, ranging from simple to sophisticated spoofing scenarios derived from publicly available TEXBAT data recordings.

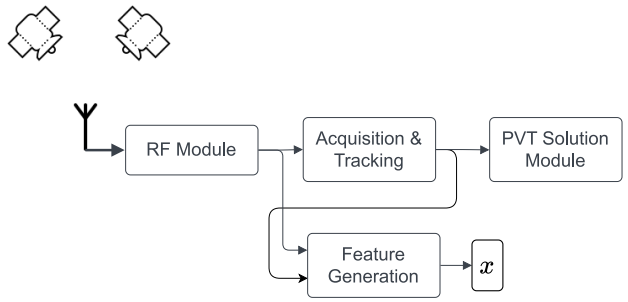


FIGURE 1. Block diagram of a typical single antenna GNSS receiver and our feature generation module.

These contributions collectively advance the field of GNSS spoofing detection by introducing a robust and adaptable model capable of detecting a wide range of spoofing attacks, including zero-day attacks, and providing practical solutions for real-world applications.

The structure of the paper is as follows: Section II provides essential background information, specifically details on feature computation, discussion on adversary threat model and highlighting the various stages of carrying out induced spoofing. In Section III, we delve into the proposed model architecture, the training algorithm, and the three distinct detection models. A comprehensive and in-depth experimental evaluation is presented in Section IV, encompassing the evaluation of supervised ML models under the classical and leave-one-out training strategy. This is followed by an extensive assessment of the proposed zero-day detectors, conducted on the entire TEXBAT dataset. Section V presents conclusion, limitations, and future work.

II. BACKGROUND

A. GPS RECEIVER

From this point forward, our discussion will center on the Global Positioning System (GPS) [2] as the primary subject for signal modeling and analysis. Specifically, our research focuses exclusively on single-antenna-based GPS receivers. A typical GPS receiver comprises three essential computational modules, collectively responsible for extracting data from the received GPS signals and utilizing this information to estimate the receiver's time and positional coordinates [33]. The structural overview of such a receiver is depicted in Figure 1. The first Radio Frequency (RF) module, plays a pivotal role in down-converting the incoming GPS signal to an Intermediate Frequency (IF), thereby enabling subsequent modules to process it effectively. Additionally, the RF module is responsible for signal amplification and quantization. The acquisition module of the GPS receiver utilizes the PRN codes of all GPS satellites to determine which satellites are currently in the receiver's line of sight. This determination is made by comparing the received power of each satellite signal against a predefined threshold. Once a satellite is identified, the receiver estimates its code delay and Doppler shifts. For all the identified satellites, the tracking module utilizes these estimates to perform parallel correlation between the incoming signal and the synchronized PRN codes. This parallel processing is followed by both coherent and incoherent integration, resulting in digital samples. These digital samples are subsequently used by the PVT solution module to estimate the receiver's position, velocity, and the current time.

B. TRACKING STAGE SIGNAL MODEL

The feature set used in this work is generated from the output of the RF module and output of the correlators present in the tracking module of a typical GPS receiver, thus, the signal model at these points is discussed next.

Considering a spoof free scenario, the genuine complex-valued signal $r_g(t)$, coming from a specific satellite, appearing at the RF module output can be written as:

$$r_g(t) = \sqrt{P_g}\, D(t-\tau_g)\, C(t-\tau_g)\, \exp(j\phi_g), \tag{1}$$

where P_g denotes power of the genuine signal, t represents time, $D(t) = \pm 1$ represents the BPSK-modulated navigation data, and $C(t)$ denotes the satellite specific BPSK-modulated PRN spreading code. τ_g and ϕ_g are the code delay and carrier phase (in radians), respectively. Note that P_g, τ_g, and ϕ_g are all time-varying, however, for notational simplicity, their temporal dependence notation is suppressed here. Without loss of generality, we can further simplify by assuming $D(t) = 1$. After passing through the tracking module, the signal at the output of tracking correlators is given by [12]

$$\begin{aligned} I_d &= \sqrt{P_g}\, R(dT_c)\, \cos(\phi_g) + \zeta_d^I, \\ Q_d &= \sqrt{P_g}\, R(dT_c)\, \sin(\phi_g) + \zeta_d^Q, \end{aligned} \tag{2}$$

here the tracking correlators' output is complex-valued, with the In-phase and Quadrature components denoted as I_d and Q_d, respectively. A single PRN code chip duration is denoted by T_c, d is unitless and (as dT_c) is used to represent code delay for each of the multiple correlators present in the receiver. For instance, for a three correlator system, d is set as $d = 0$ for prompt, $d > 0$ for late, and $d < 0$ for early correlators. The noise components in I and Q channels are denoted by ζ_d^I and ζ_d^Q, respectively, which contain zero-mean thermal Gaussian noise, with a constant noise spectral density N_0. Finally, $R(\cdot)$ denotes the ideal auto-correlation function of a BPSK modulated code, also shown in Fig. 2 can be written as

$$R(dT_c) = \begin{cases} 1 - |dT_c|/T_c, & |dT_c| \le T_c, \\ 0, & |dT_c| > T_c. \end{cases} \tag{3}$$

The I_d and Q_d components from (2) are assumed to be independent Gaussians with statistics [34]

$$\begin{aligned} \mu_I &= \sqrt{P_g}\, R(dT_c)\, \cos(\phi_g), \quad \mu_Q = \sqrt{P_g}\, R(dT_c)\, \sin(\phi_g) \\ \sigma_I^2 &= \sigma_Q^2 = \sigma_0^2 = \frac{1}{2T_{int}(C/N_0)}, \quad \sigma_{IQ}^2 = 0, \end{aligned} \tag{4}$$

here μ_I, μ_Q, σ_I^2, σ_Q^2 are the statistics of I and Q components given in (2), and their covariance is σ_{IQ}^2. σ_0^2 represents the noise variance at the correlators' output, and T_{int} is the coherent integration period of the output signal.

In the presence of a spoofed signal, the tracking correlators' output signal (2) will have an additional component corresponding to this spoofed signal, thus the new signal model will become

$$\begin{aligned} I_d &= \sqrt{P_g}\, R(dT_c)\, \cos(\phi_g) + \sqrt{\eta P_g}\, R(dT_c - \Delta\tau) \\ &\quad \cos(\phi_g + \Delta\phi) + \zeta_d^I, \\ Q_d &= \sqrt{P_g}\, R(dT_c)\, \sin(\phi_g) + \sqrt{\eta P_g}\, R(dT_c - \Delta\tau) \\ &\quad \sin(\phi_g + \Delta\phi) + \zeta_d^I, \\ \psi_d &= I_d + Q_d i, \end{aligned} \tag{5}$$

here, the spoofed signals' power advantage over the genuine signal is denoted by $\eta = P_s/P_g$, which is measured over the integration period. Similarly, $\Delta\tau$, and $\Delta\phi$ are the code and carrier phase offsets of the spoofed signal w.r.t. the genuine one. The overall complex-valued output is denoted by ψ_d.

C. THREAT MODEL

Typically, an attacker aiming to compromise a system requires in-depth knowledge of its modules and subsystems. The greater the information they possess, the higher the likelihood of a successful attack. In GNSS spoofing attacks, the civilian L1 C/A channel is an open standard [33], providing attackers with all the necessary details to generate such signals. To execute an attack, an adversary can utilize commercially available Universal Software Radio Peripheral (USRP) devices for the RF part of the GNSS receiver, while implementing the remaining transmit-receiver chain in software. Moreover, the target receiver should be within the transmission range of the attacker's antenna. With two USRP devices (TX and RX) and a capable computer system, an attacker could execute an 'Induced Spoofing' attack on the target receiver. This type of attack represents an intermediate level attack, wherein the attacker strategically adjusts code phase and power levels subtly to avoid unlocking the tracking loop of the target receiver. The entire process of such an attack is illustrated in Fig. 2 and is detailed below:

1) To initiate the attack, the attacker first estimates the speed and location of the target receiver. It also needs to estimate the number of satellites in view and capture their navigation signals at the target receiver's antenna. Armed with this information, the spoofer generates a low-powered signal centered at the same frequency (center frequency plus Doppler shift). However, this signal intentionally lags the genuine signal in code phase by more than 2 chips, as depicted in Fig. 2 (a). Subsequently, the spoofer incrementally adjusts its code phase to gradually approach the genuine signal.
2) The spoofer continues to adjust its code delay until it synchronizes with the genuine signal, typically achieving synchronization within 0.5 chips, as depicted in Fig. 2 (b) and (c). Once synchronized, the spoofer begins to increase its signal power to slightly surpass that of the genuine signal. Through careful alterations of its code phase, it starts to carry off the resultant tracking correlator's peak, as illustrated in Fig. 2 (d) and (e). This phase of the attack is where the correlation peak of the tracking correlators experiences the significant distortion.
3) The spoofer continues carrying off the correlation peak until it is at least 2 chips ahead of the genuine one, as depicted in Fig. 2 (f). Finally, the spoofer gradually reduces its signal power to nominal levels, thus completing the takeover of the target receiver's tracking loop.

In intermediate-level attacks, as discussed above, the attacker maintains the initial phase offset between the genuine and spoofed signals throughout the attack [22]. This results in substantial distortion in correlation peak symmetry, as shown in Fig. 2. SQM metrics are specifically designed to detect and capture this distortion, playing a crucial role in identifying induced spoofing attacks. However, in cases where the attacker possesses precise knowledge of the receiver antenna's position and a detailed fading model between the spoofer and the target antenna, they can execute a more sophisticated and subtle attack where the attacker aligns the carrier phase of the spoofed signal precisely with the authentic one, resulting in minimal correlation peak distortion [23]. Such an attack, termed 'sophisticated,' is challenging to execute unless the attacker has physical access or can approach the target receiver's antenna closely [12], [17]. Instances such as shipment vehicle monitoring, asset tracking, and smart traffic management could potentially face this kind of attack. In this study, apart from assessing various intermediate-level attack scenarios, we also evaluate our proposed detectors against a sophisticated-level attack.

D. FEATURE COMPUTATION

In this work, we utilized the widely-used open-source single antenna based software receiver called FGI-GSRx [33]. The receiver is MATLAB based and gives full access to the received signal at various stages of the signal processing pipeline. All 7 features examined in this paper can be obtained in real-time from the output of the RF block and tracking block of a GPS receiver. To compute each feature, we average over a 20 ms time window. The extraction methodology is described in detail below.

1) RECEIVED POWER

In case of the civilian L1 GPS band, most of the signal power is concentrated in a 2 MHz band around the L1 carrier frequency of 1575.42 MHz, thus we filter the RF output signal using a 2 MHz bandwidth filter. Let $y_{RF}[n]$ be the complex-valued baseband samples at the output of the receivers' RF block. We pass it through a low pass filter to get a filtered version $\tilde{y}_{RF}[n]$, then for a given time interval, received power (in dBW) can be computed as

$$P[k] \triangleq 10\, log_{10}\left(\frac{1}{N}\sum_{n=(k-1)N+1}^{kN} |\tilde{y}_{RF}[n]|^2\right) \tag{6}$$

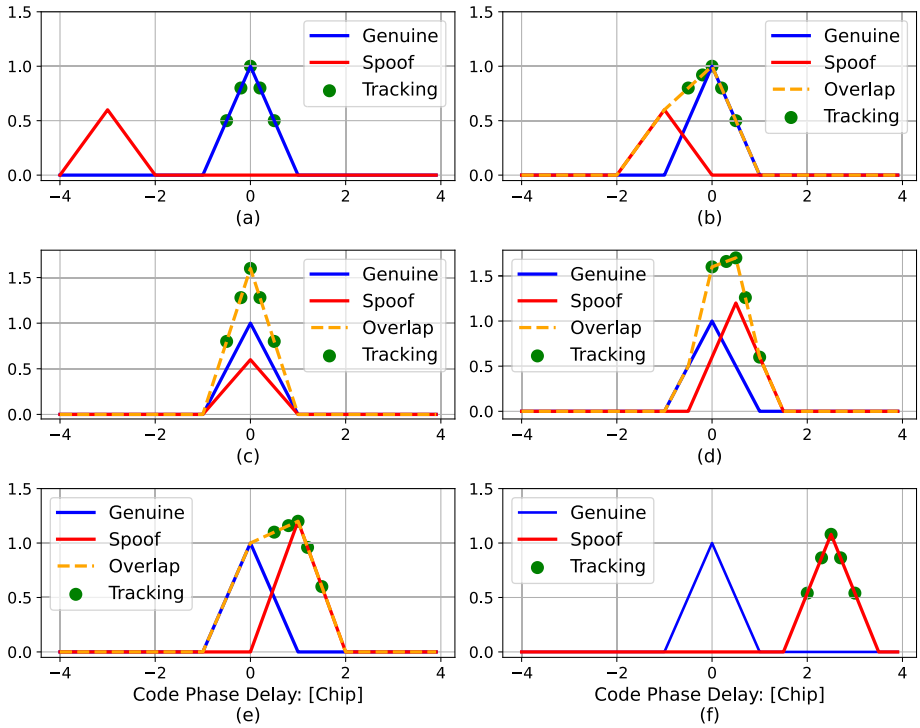


FIGURE 2. Various stages of a typical induced spoofing attack observed within the correlators of the tracking module.

here N is the number of samples in a 20 ms time window. The initial power value (in dB) is subtracted from the entire vector to create a relative received power feature.

2) CARRIER TO NOISE RATIO

The C/N_0 is a key metric to measure the received GPS signals' strength. However, direct measurement of C/N_0 is not possible and requires estimation. To estimate this metric, we use the well-known Narrow-band Wide-band Power Ratio (NWPR) method [33]. The reason for considering both the received power and C/N_0 is to enable the trained classifier to distinguish between genuine interference and spoofing cases, as discussed in [21].

3) SIGNAL QUALITY MONITOR

In order to capture the correlation peak distortion during the spoofed and authentic signal interaction, we use 5 different SQM metrics as input features. These metrics are given below:

- Ratio Metric [15]

$$m_{ratio} = \frac{I_{-d} + I_{+d}}{I_P}, \tag{7}$$

- Delta Metric [15]

$$m_{delta} = \frac{I_{-d} - I_{+d}}{I_P}, \tag{8}$$

- Early Late Phase (ELP) Metric [16]

$$m_{elp} = \tan^{-1}\left(\frac{Q_{-d}}{I_{-d}}\right) - \tan^{-1}\left(\frac{Q_{+d}}{I_{+d}}\right), \tag{9}$$

- Symmetric Differences [12]

$$m_{sd} = \frac{|\psi_{-d} - \psi_{+d}|}{\sigma_{N_0}}, \tag{10}$$

- Manfredini Metric [21]

$$m_{fred} = \frac{|E_x - L_x|}{|\psi_P|}. \tag{11}$$

Here the variables $I_{\pm d}$, $Q_{\pm d}$, and $\psi_{\pm d}$ are given in (5) with $d = 0.5$. I_P and ψ_P are the respective prompt correlator values. m_{fred} is computed using 9 correlators evenly spaced between $d = [-0.1016, 0.1016]$, with L_x and E_x being the linear combination of complex values from late and early correlator fingers, respectively. Finally, σ_{N_0} is the standard deviation of ψ_{-2} during the spoof free case.

While the inclusion of received power and C/N_0 in the feature set can aid the model in discerning between regular RFI and spoofing attacks, this aspect doesn't constitute the primary focus of our research [21]. These features enable our model not only to detect high-powered spoofer promptly upon their arrival at the receiver's antenna but also to handle matched power spoofing attacks. In these cases, the spoofer maintains a slight increase in received power, making it detectable when combined with other correlation peak distortion monitors. In addition to the five SQM metrics in our

feature set, we explored three other metrics–S-curve bias (SCB) [35], slope-based metrics [18], and WSCM-based metric [19]. However, they were excluded from our final feature selection due to being highly correlated to at least one of our chosen basic features, and thus did not provide any significant performance enhancements.

The rationale behind choosing these five distinct SQM metrics rests on their complementarity, each contributing uniquely to the detection process. For instance, when the metric m_{elp} exhibits a large value, m_{delta} and m_{ratio} tend to have smaller values, and vice versa, as evidenced in previous studies [17]. The metric m_{sd} quantifies the absolute difference between early and late correlators scaled by the standard deviation of noise (in the absence of spoofing). Notably, it tends to produce high values both during and after the complete takeover of the receiver's correlator peak. Lastly, m_{fred} is particularly sensitive to small distortions that occur during the initial stages of correlator peak pull-off. This sensitivity arises from its utilization of correlator fingers that are in close proximity to the prompt signal.

E. THE BASE LEARNER - FCNN

The foundational component of the proposed model, as delineated in Section III, is a Fully Connected Neural Network (FCNN), also commonly referred to as an Artificial Neural Network (ANN). This neural network architecture comprises several essential layers, including an input layer, multiple hidden layers, and an output layer, as visually represented in Fig. 3. The input layer serves as the entry point for the feature vector, accepting it as input and making it available for processing by neurons in subsequent layers. Each hidden layer plays a crucial role in this network's operation by taking the outputs from neurons in the preceding layers, applying non-linear transformations to them generating its own set of outputs. This process, known as forward propagation, continues sequentially through the network's layers until it reaches the final layer, which is responsible for producing the network's final output.

The basic element of an FCNN is a neuron which typically has three components associated with it, a weight vector, a bias value, and an activation function. Let a_i^l be the m^{th} neuron of l^{th} layer of an ANN. The output of this neuron is generated as

$$s_m^l = f(\mathbf{w}_m^l\, \mathbf{a}^{l-1} + b_m), \tag{12}$$

where $\mathbf{a}^{l-1} = \left[a_1^{l-1}, a_2^{l-1}, \ldots, a_n^{l-1}\right]^\top$ is a column vector containing outputs previous layers' neurons, $\mathbf{w}_m^l = \left[w_{m1}^l, w_{m2}^l, \ldots, w_{mn}^l\right]$ is a row vector containing the linear coefficients/weights of the m^{th} neuron, n is the number of neurons in the previous layer, b_m is the bias term, and f is a point-wise non-linear activation function. These equations can be rewritten for an entire layer of neurons as

$$\mathbf{a}^l = f(\mathbf{W}^l\, \mathbf{a}^{l-1} + \mathbf{b}^l). \tag{13}$$

Here $\mathbf{W}^l \in \mathbb{R}^{m\times n}$ is the coefficient matrix for the l^{th} layer, $\mathbf{a}^{l-1} \in \mathbb{R}^n$ contains the outputs of previous layer, $\mathbf{b}^l \in \mathbb{R}^m$ is the bias vector. Output of the final layer is used to construct a loss function which evaluates the network's performance in its intended task. This loss function is optimized by adjusting the model parameters (weights and biases). In practice, these parameters are often challenging to solve analytically. However, iterative optimization algorithms, such as gradient descent, are adept at efficiently approaching local optimal solutions. Keen reader is referred to [36] for additional information.

III. PROPOSED SPOOF DETECTION MODEL

As previously discussed, under the supervised training framework, models are typically trained using all possible classes or categories of data they are expected to recognize. This extensive training equips the models to identify new samples that may not be identical to their training counterparts but share similar features with them. However, such models may struggle to reliably recognize entirely dissimilar samples [37]. In the context of spoofing detection, our objective is to create and train a model that not only performs well on the data it is trained on but also exhibits robustness in recognizing samples it has never encountered before. Therefore, in the case of a novel attack scenario, our model should possess the capability to flag it as potentially malicious. Representation learning / profiling-based training frameworks offer a promising approach in this context [38]. These frameworks involve training models using data from a single known class (in our case, genuine GPS data) and subsequently using these models to classify whether a test sample shares a similar distribution to the training set or not. Such detectors are often referred to as *zero-day detectors* because they excel at detecting samples originating from unknown distributions.

To this end, in this section, we present the model architecture, discuss the forward pass, cover loss computation, outline training algorithm, and explore three detection frameworks.

A. MODEL ARCHITECTURE

The proposed model consists of three interconnected FCNNs: the Encoder ($\mathcal{E}_{\boldsymbol{\phi}}$), Decoder ($\mathcal{D}_{\boldsymbol{\theta}}$), and Critic ($\mathcal{C}_{\boldsymbol{\varphi}}$) as shown in Fig. 3. Architecturally, the encoder and decoder networks operate in a VAE configuration [39], while the decoder and critic networks function as a Wasserstein GAN termed (WGAN) [40]. It's important to note that the decoder network is shared between both branches of the model. The data flow through each network is discussed below.

1) THE ENCODER

The encoder model takes as input the feature vector $\mathbf{x}$, which contains feature values at a specific time instance (as detailed in Section II), along with a one-hot-encoded binary class label $\mathbf{c}$ indicating whether the feature is from a static ($[1, 0]^\top$) or dynamic ($[0, 1]^\top$) receiver. This class label helps in improving the encoder's capability to map samples from both classes closer to the origin of the latent space. The encoder produces

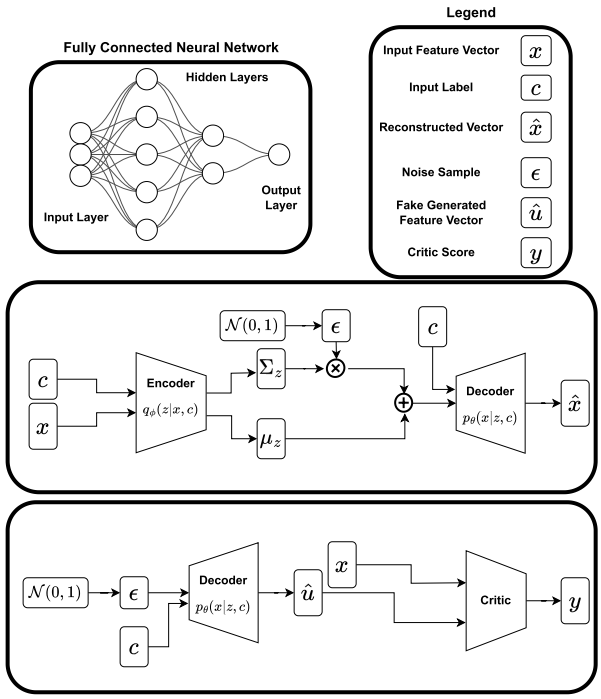


FIGURE 3. **A complete diagram of the proposed model.**

two output vectors of equal sizes: a mean vector $\boldsymbol{\mu}_\mathbf{z}$ and a variance vector $\boldsymbol{\Sigma}_\mathbf{z}$, both sized according to the dimensions of the latent space. These vectors collectively define a latent probability distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$, parameterized by the encoder. During training, the objective is to enable the encoder to map its inputs into a continuous latent space, where each input corresponds to a region in the latent space rather than a single point [39]. This facilitates the subsequent generation of samples by the decoder (discussed below), which are not exact replicas of their input counterparts.

2) THE CRITIC

In WGANs, the traditional discriminator found in vanilla GANs is substituted with a critic. Unlike a binary classifier, the critic functions as an evaluator, providing high scores for real samples and low scores for fake ones generated by its corresponding generator network, which in this case is our decoder. The term 'adversarial' in GAN signifies that during the network training phase, the critic and generator networks engage in a competitive process. The critic model takes as input both real data samples $\mathbf{x}$ and generated fake data samples $\hat{\mathbf{u}}$, assigning a score to each sample. Its training objective is to maximize the predictive difference between real and generated fake data samples. This adversarial interplay between the critic and generator networks is instrumental in refining the model's ability to generate realistic data representations, and consequently helping critic to be more, critical.

3) THE DECODER

While the encoder and critic networks do not directly interact, they both utilize the shared decoder network, each with a distinct input approach:

- When collaborating with the encoder, the decoder receives the latent variable $\mathbf{z}$, sampled from the latent probability distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$ using the reparameterization trick [39]:

$$\mathbf{z} = \boldsymbol{\mu}_\mathbf{z} + \boldsymbol{\epsilon} \odot \boldsymbol{\Sigma}_\mathbf{z}, \tag{14}$$

  where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ represents element-wise multiplication. This enables gradient backpropagation through the sampling process into the encoder model while preserving the probabilistic nature of the sampling. Using $\mathbf{z}$ and the class label $\mathbf{c}$, the decoder is trained to reconstruct the original input $\mathbf{x}$ by sampling from the distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}, \mathbf{c})$ (parameterized by the decoder).
- When operating alongside the critic, the input $\mathbf{z}$ to the decoder is generated similarly to the reparameterization trick in (14), but with $\boldsymbol{\mu}_\mathbf{z} = \mathbf{0}$ and $\boldsymbol{\Sigma}_\mathbf{z} = \mathbf{1}$. In this context, the decoder's objective is to learn the real data distribution $p_g(x)$.

In our proposed model, the encoder, decoder, and critic networks collaborate harmoniously to achieve the overarching goal of robust spoof detection. Training solely on authentic data enhances the encoder and decoder's proficiency in reconstructing samples conforming to the training data distribution. Simultaneously, the critic becomes adept at evaluating the authenticity of both real and generated data samples. Once trained, the model can be used in three distinct ways to detect spoofing attacks, which are discussed in Section III-III-C.

B. MODEL LOSS COMPUTATION AND TRAINING ALGORITHM

1) VAE LOSS

Consider a data vector $\mathbf{x}$ and its label $\mathbf{c}$, a probabilistic encoder encodes these inputs into the latent vector $\mathbf{z}$ according to the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$, and a probabilistic decoder decodes $\mathbf{z}$ using $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}, \mathbf{c})$ into $\hat{\mathbf{x}}$. The encoder and decoder networks are trained together by maximizing the following loss function

$$\mathcal{L}_{VAE}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{z}, \mathbf{c}) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{c})}(log\ p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}, \mathbf{c})) - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})||p_z(\mathbf{z})). \tag{15}$$

Equation (15) is also called the variational lower bound [39]. The first term in (15) is the log likelihood function and the second term is the latent loss which uses Kullback-Leibler Divergence (KLD) between the learned latent distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$ and a prior distribution $p_z(\mathbf{z})$. Here, the KLD is used as a regularizer to induce some structure onto the latent distribution. In its current form, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$ is intractable and the KLD between the latent and the selected prior needs to be estimated [39]. However, if we assume $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{c})$ to be Gaussian with an approximately diagonal covariance, and $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ to be a unit Gaussian, the KLD can be computed without estimation [39]. Furthermore, assuming a Laplacian $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}, \mathbf{c})$, the (15) can be written as

$$\mathcal{L}_{VAE}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}, \mathbf{c}) \simeq \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{J}(1 + log(\sigma_{ij}^2) - \mu_{ij}^2 - \sigma_{ij}^2)$$

$$-\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i-\hat{\mathbf{x}}_i\|_1, \quad (16)$$

here N is the total number of batch samples, J is the dimension of $\mathbf{z}$, μ_{ij} is the j^{th} element of $\boldsymbol{\mu}_i$, σ_{ij} is the j^{th} element of $\boldsymbol{\Sigma}_i$, and $\|\cdot\|_1$ is the vector ℓ_1-norm. Here we chose $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{c})$ to be Laplacian to enable the reconstruction error to be robust to outlier samples which are often found in real datasets [41].

2) WGAN LOSS

In the standard GAN, the loss function relies on the Jensen-Shannon Divergence (JSD) between real p_r and generated p_g data distributions. However, JSD has limitations; it fails to effectively measure the distance between distributions when there is minimal to no overlap between them [42]. This can lead to the discriminator becoming perfect, resulting in the vanishing gradient problem during training. To circumvent this issue and ensure stable training, we adopt the WGAN framework.

WGAN replaces the JSD of the vanilla GAN with the Earth Mover Distance (EMD), also known as the Wasserstein Distance (WD). EMD, a member of the Integral Probability Metrics (IPM) family, quantifies the effort required to transform one distribution into another by redistributing mass from one region to another. Unlike JSD, WD is a smooth and effective distance metric, even when the distributions being compared exhibit minimal or no overlap. Formally, the WD between p_r and p_g is given by

$$\mathrm{WD}(p_r,p_g)=\inf_{\gamma\sim\prod(p_r,p_g)}\mathbb{E}_{(x,y)\sim\gamma}\left[\|x-y\|\right], \quad (17)$$

where $\prod(p_r,p_g)$ is the set of all possible joint probability distributions with $p_r(x)$ and $p_g(y)$ as its marginal distributions. Informally, $\gamma(x,y)$ can be considered a transport plan, i.e., the amount of mass required to be moved from y to x in order to transform p_g into p_r. Working with (17) is difficult as it is not possible to exhaust all possible distributions of $\prod_{(p_r,p_g)}$ [40]. Instead, authors propose to solve (17) using the functional format given below

$$\mathrm{WD}(p_r,p_g)=\sup_{\|f\|_L\le 1}\mathbb{E}_{x\sim p_r}[f(x)]-\mathbb{E}_{x\sim p_g}[f(x)], \quad (18)$$

where the function f is 1-*Lipschitz* (i.e., $|\frac{d(f)}{dx}|\le 1$) and the supremum is taken over it.

In the WGAN model, the critic's output is a real number rather than a probability, with higher values assigned to real samples. Consequently, the WGAN critic is trained to maximize the difference in its predictions between genuine data and generated fake data samples. To encourage the critic to learn a 1-*Lipschitz continuous* function, a critical aspect of WGAN training, the learned weights are constrained to remain within a narrow range, typically $[-0.01, 0.01]$, following each training batch. The critic model is trained by maximizing the following loss function:

$$\mathcal{L}_{critic}=\max_{\boldsymbol{\varphi}\in F}\mathbb{E}_{X\sim p_r(x)}[\mathcal{C}_{\boldsymbol{\varphi}}(x)]-\mathbb{E}_{Z\sim p_z(z)}[\mathcal{C}_{\boldsymbol{\varphi}}(\mathcal{D}_{\boldsymbol{\theta}}(z,c))], \quad (19)$$

where, F is the set of 1-*Lipschitz continuous* functions, $\mathcal{C}_{\boldsymbol{\varphi}}$ is the critic network, and $\mathcal{D}_{\boldsymbol{\theta}}$ is the decoder network. Although WGAN has been shown to stabilize training, the weight clipping deployed in the critic update of WGAN greatly lowers its learning capacity and often leads to convergence failure. To solve this issue, instead of gradient clipping, we augment (19) with a Gradient Penalty (GP) to enforce the Lipschitz constraint on the critic [43]. The resulting critic loss becomes

$$\mathcal{L}_{critic}=\max_{\boldsymbol{\varphi}\in F}\mathbb{E}_{X\sim p_r(x)}[\mathcal{C}_{\boldsymbol{\varphi}}(x)]-\mathbb{E}_{Z\sim p_z(z)}[\mathcal{C}_{\boldsymbol{\varphi}}(\mathcal{D}_{\boldsymbol{\theta}}(z,c))]-\lambda\,\mathbb{E}_{\tilde{x}}\left[(\|\Delta\mathcal{C}_{\boldsymbol{\varphi}}(\tilde{x})\|_2-1)^2\right] \quad (20)$$

where $\tilde{x}$ is uniformly sampled from the line segment joining points sampled from p_r and p_g, respectively, $\|\cdot\|_2$ is the vector ℓ_2-norm and λ is the GP controlling parameter. WGAN-GP has been shown to learn a better parameter distribution w.r.t. WGAN, and has been used as the default in many GAN loss variants.

Finally, the generator (decoder) in WGAN is trained to generate samples as close to the real ones as possible, in order to fool the critic into giving fake samples a higher score. The loss function to ensure this is given by

$$\mathcal{L}_{Decoder}=\max_{\boldsymbol{\phi}}\mathbb{E}_{Z\sim p_z(z)}\left[\mathcal{C}_{\boldsymbol{\varphi}}(\mathcal{D}_{\boldsymbol{\theta}}(z,c)\right] \quad (21)$$

The fusion of VAE and GAN architectures offers the advantage of harnessing the strengths of both paradigms. By doing so, we can capitalize on the precise reconstruction capabilities of VAE models while leveraging the feature learning prowess of the GAN architecture. This synergy allows us to acquire a rich representation of the underlying data, resulting in a more robust and effective model for our spoof detection task.

To consolidate the training process, we outline the algorithmic steps for the model depicted in Fig. 3 in Algorithm 1. In this algorithm, the $\mathcal{C}_{\boldsymbol{\varphi}}$ model is updated κ times for each update of $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{D}_{\boldsymbol{\theta}}$ models. This repeated updating of $\mathcal{C}_{\boldsymbol{\varphi}}$ ensures its convergence, consequently yielding accurate gradients for the $\mathcal{D}_{\boldsymbol{\theta}}$ model in (21) [40]. While we do not provide a formal proof of convergence for the training algorithm, we showcase the empirical loss curves for training the model with $\kappa = 5$ in Fig. 4. The plotted curves demonstrate consistent improvement across all component losses in the initial 60 epochs, prompting us to prolong the training process to ensure thorough model parameter stability. For detailed insights into the specific model hyper-parameters, please refer to Section IV-C1.

C. DETECTION METHODS

In this section, we discuss the development of detection methods built on the premise of recognizing similarity between

Algorithm 1 The Training Algorithm

Input: Data samples $\mathbf{X}$, labels $\mathbf{C}$, epochs, λ, model training ratio κ, total batches nB, models $\mathcal{E}_{\boldsymbol{\phi}}, \mathcal{D}_{\boldsymbol{\theta}}, \mathcal{C}_{\boldsymbol{\varphi}}$.

Initialization: Initialize $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and $\boldsymbol{\varphi}$, the model parameters of $\mathcal{E}_{\boldsymbol{\phi}}, \mathcal{D}_{\boldsymbol{\theta}}, \mathcal{C}_{\boldsymbol{\varphi}}$,
for *t in epochs* **do**
  for *b in nB* **do**
    Fetch new batches for $\mathbf{x}$, $\mathbf{c}$.
    Training the Critic $\mathcal{C}_{\boldsymbol{\varphi}}$:
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Sample $\alpha \sim \mathcal{U}(0, 1)$
    Compute: $\mathbf{y}_{\mathbf{x}} = \mathcal{C}_{\boldsymbol{\varphi}}(\mathbf{x})$, $\hat{\mathbf{u}} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{c})$, and $\mathbf{y}_{\hat{\mathbf{u}}} = \mathcal{C}_{\boldsymbol{\varphi}}(\hat{\mathbf{u}})$
    $\tilde{\mathbf{x}} = \alpha\,\mathbf{x} + (1-\alpha)\,\hat{\mathbf{u}}$
    $\mathbf{y}_{\tilde{\mathbf{x}}} = \mathcal{C}_{\boldsymbol{\varphi}}(\tilde{\mathbf{x}})$
    Compute (20) as
    $\mathcal{L}_{critic} = \mathbb{E}[\mathbf{y}_{\mathbf{x}}] - \mathbb{E}[\mathbf{y}_{\hat{\mathbf{u}}}] - \lambda\,\mathbb{E}[(\|\Delta \mathbf{y}_{\tilde{\mathbf{x}}}\|_2 - 1)^2]$
    Perform gradient backprop. through $\mathcal{C}_{\boldsymbol{\varphi}}$ using $\mathcal{L}_{critic}$,
    Update $\boldsymbol{\varphi}$, the parameters of $\mathcal{C}_{\boldsymbol{\varphi}}$ to maximize $\mathcal{L}_{critic}$.
    Training the Encoder $\mathcal{E}_{\boldsymbol{\phi}}$ and Decoder $\mathcal{D}_{\boldsymbol{\theta}}$:
    if $mod(t, \kappa) == 0$ **then**
      Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      Compute:
      $\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z = \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{x}, \mathbf{c})$
      $\mathbf{z} = \boldsymbol{\mu}_z + \boldsymbol{\epsilon} \odot \boldsymbol{\Sigma}_z$
      $\hat{\mathbf{x}} = \mathcal{D}_{\boldsymbol{\theta}}(\mathbf{z}, \mathbf{c})$
      $\hat{\mathbf{u}} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathbf{c})$
      $\mathbf{y}_{\hat{\mathbf{u}}} = \mathcal{C}_{\boldsymbol{\varphi}}(\hat{\mathbf{u}})$
      Compute $\mathcal{L}_{VAE}$ using (16)
      Compute (21) as $\mathcal{L}_{Decoder} = \mathbb{E}[\mathbf{y}_{\hat{\mathbf{u}}}]$
      Compute $\mathcal{L}_{total} = \mathcal{L}_{VAE} + \mathcal{L}_{Decoder}$
      Perform gradient backprop. through $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{D}_{\boldsymbol{\theta}}$ using $\mathcal{L}_{total}$
      Update $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, the parameters of $\mathcal{E}_{\boldsymbol{\phi}}$ and $\mathcal{D}_{\boldsymbol{\theta}}$, respectively to maximize $\mathcal{L}_{total}$.

Output: The trained $\mathcal{E}_{\boldsymbol{\phi}}, \mathcal{D}_{\boldsymbol{\theta}}, \mathcal{C}_{\boldsymbol{\varphi}}$ models.

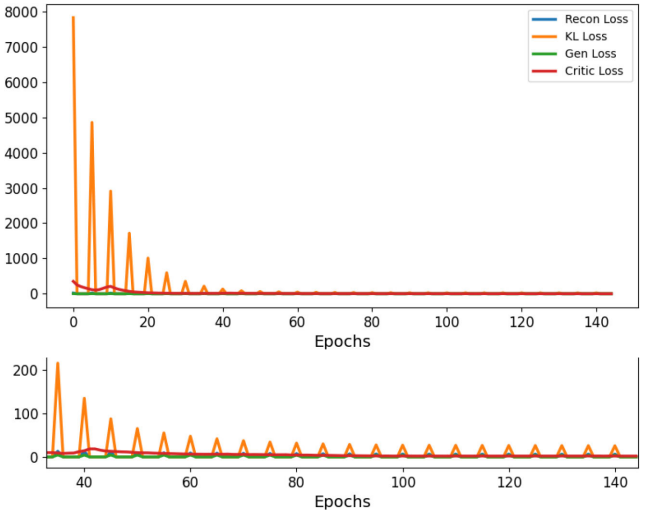


FIGURE 4. **Convergence graph for model training using algorithm 1.**

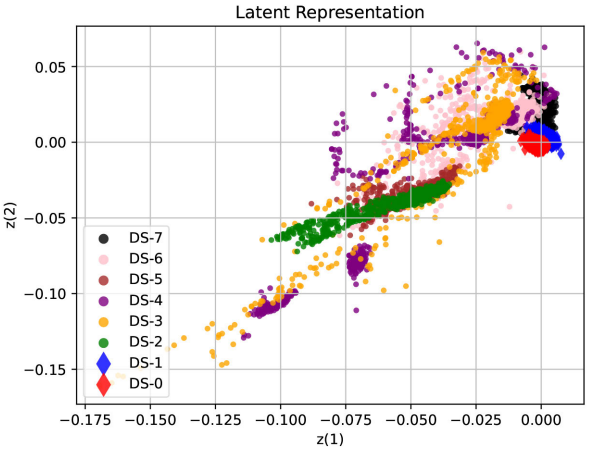


FIGURE 5. **Latent representations generated by the Encoder $\mathcal{E}_{\boldsymbol{\phi}}$.**

test samples and those used for model training. Specifically, our goal is to equip the detection model with the capability to discern whether an input sample closely resembles the trained data or deviates significantly from it. To achieve this, we employ the computation of three distinct test statistics. With the careful selection of appropriate thresholds, these statistics serve as the basis for flagging samples as either genuine or spoofed, resulting in the creation of three separate detectors. The subsequent sections will elaborate on the computation of these statistical measures.

1) THE ℓ_2 CRITERION BASED DETECTOR D_{ℓ_2}

The encoder network, denoted as $\mathcal{E}_{\phi}$, plays the central role in our first detection model. Its primary objective is to map genuine input features into a latent space that is explicitly centered at the origin and symmetrically distributed around it, a constraint enforced through the KLD in (15). Consequently, spoofed samples deviate significantly from this centered origin. To establish a baseline for our detection, denoted as $\mathbb{E}[\boldsymbol{\mu}_{z_g}]$, we calculate the expected value of the mean vector $\boldsymbol{\mu}_{z_g}$, where $\boldsymbol{\mu}_{z_g}, \boldsymbol{\Sigma}_{z_g} = \mathcal{E}_{\boldsymbol{\phi}}(\mathbf{X}_g, \mathbf{C}_g)$. Here, $\mathbf{X}_g$ represents the genuine training samples, and $\mathbf{C}_g$ signifies their corresponding labels.

Our test statistic for a given test sample $\bar{\mathbf{x}}$ is computed as $\zeta_{\ell_2}(\bar{\mathbf{x}}) = \|\boldsymbol{\mu}_{z_{\bar{\mathbf{x}}}} - \mathbb{E}[\boldsymbol{\mu}_{z_g}]\|_2$. Our experimental results show that $\mathbb{E}[\boldsymbol{\mu}_{z_g}]$ closely approximates the origin. Subsequently, for a predefined threshold ρ, if $\zeta_{\ell_2} < \rho$, the sample is classified as genuine; otherwise, it is deemed spoofed. This detection method, founded on the ℓ_2 distance, is denoted as D_{ℓ_2}. To visually illustrate the latent representations generated by the trained $\mathcal{E}_{\phi}$ across all datasets (as shown in Table. 2),

we selected 1000 random samples from each dataset and present their representations in Fig. 5. Notably, samples originating from genuine static and dynamic training datasets (DS-0 and DS-1) are mapped closer to the origin due to the reduction of KLD component in (15). Conversely, depending on their similarity to the training samples, samples from other datasets (i.e., attack samples) are mapped closer or further from the origin. For instance, DS-2 and DS-5 attack dataset samples are mapped away from the origin, indicating their significant dissimilarity from the genuine training samples, as they originate from high-powered spoofing scenarios. These observations validate the appropriateness of distance from the origin as a viable test statistic for spoof detection.

2) THE RECONSTRUCTION ERROR BASED DETECTOR D_{rec}

The decoder network, denoted as $\mathcal{D}_\theta$, serves a pivotal role in our second detection model. It is trained to generate samples that closely resemble those from the genuine sample distribution, a goal achieved by maximizing the objective functions in (16) and (21). When a test sample $\mathbf{x}$, along with its corresponding test label $\mathbf{c}$, is passed through the $\mathcal{E}_\phi$ model, it generates a latent variable $\mathbf{z}$ (computed using (14)). Simultaneously, $\mathbf{z}$ and $\mathbf{c}$ are passed through $\mathcal{D}_\theta$ to produce the reconstructed sample $\hat{\mathbf{x}}$. In cases where the test sample is genuine, the model should be able to reconstruct it well. Conversely, for spoofed samples, the model is expected to struggle in replicating them since it was not exposed to such samples during training.

Under this premise, we calculate a test statistic for a given test sample $\bar{\mathbf{x}}$ and its reconstruction $\hat{\bar{\mathbf{x}}}$ as $\zeta_{rec} = \|\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}\|_1$. Subsequently, employing a predefined threshold ρ, if $\zeta_{rec} < \rho$, the test sample is classified as genuine; otherwise, it is designated as spoofed. This detection method, based on the reconstruction error, is termed D_{rec}.

3) THE CRITIC SCORE BASED DETECTOR D_{cr}

The critic network, denoted as $\mathcal{C}_{\boldsymbol{\varphi}}$, plays the central role in our third detection model. It is trained by maximizing the objective in (20), aided by a proficient generator (decoder). This training equips the critic to effectively distinguish between genuine samples, which were part of its training set, and spoofed ones, which it has never encountered before. When a test sample $\bar{\mathbf{x}}$ is presented to the critic, it generates a score, denoted as $\zeta_{cr} = \mathcal{C}_{\boldsymbol{\varphi}}(\bar{\mathbf{x}})$. This score is high for genuine test samples and low for spoofed ones, making it a robust statistic for spoof detection. Notably, unlike ζ_{ℓ_2} and ζ_{rec}, computing ζ_{cr} does not necessitate knowledge of the test sample's class label. Consequently, a predefined threshold ρ is applied to this score. If $\zeta_{cr} > \rho$, the test sample is classified as genuine; otherwise, it is flagged as spoofed. This detection method, is named D_{cr}.

4) THRESHOLD COMPUTATION

Selecting an appropriate threshold is crucial for the detection performance of the models. The threshold ρ is determined based on the acceptable False Positive Rate (FPR) tolerance, typically provided in the detector's design specifications. After training the models, we calculate the three test statistics (ζ_{ℓ_2}, ζ_{rec}, and ζ_{cr}) for the entire training dataset and determine their respective thresholds. These thresholds are set to correspond to the specified FPR for the training dataset. When presented with a test sample $\bar{\mathbf{x}}$, each detection criterion independently identifies it as spoofed if it surpasses its respective threshold, given as

$$\begin{aligned} D_{\ell_2} &: \zeta_{\ell_2}(\bar{\mathbf{x}}) \triangleq \|\boldsymbol{\mu}_{z_{\bar{\mathbf{x}}}} - \mathbb{E}[\boldsymbol{\mu}_{z_g}]\|_2 \geq \rho_{\ell_2}, \\ D_{rec} &: \zeta_{rec}(\bar{\mathbf{x}}) \triangleq \|\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}\|_1 \geq \rho_{rec}, \\ D_{cr} &: \zeta_{cr}(\bar{\mathbf{x}}) \triangleq \mathcal{C}_{\boldsymbol{\varphi}}(\bar{\mathbf{x}}) \leq \rho_{cr}. \end{aligned} \tag{22}$$

The detection criteria discussed above follow different pipelines to provide a detection result for a test sample $\bar{\mathbf{x}}$, i.e.,

- For $\zeta_{\ell_2}(\bar{\mathbf{x}})$, the test sample passes through the encoder network, the output is used to perform a difference operation followed by a norm computation to generate the test statistic.
- For $\zeta_{rec}(\bar{\mathbf{x}})$, the test sample first passes through the encoder, whose output is used to generate latent variable $\mathbf{z}$ (using (14)), which is passed through the decoder for sample reconstruction. The output is then used in a difference operation followed by a norm computation to generate the test statistic.
- For $\zeta_{cr}(\bar{\mathbf{x}})$, the test sample is passed through the critic network only to generate the test statistic.

Based on the above discussion and assuming that the FCNNs constituting each of these models have similar computational complexities, the test statistics can be sorted in terms of their decreasing computational requirements as follows: $\zeta_{cr}(\bar{\mathbf{x}}) < \zeta_{\ell_2}(\bar{\mathbf{x}}) < \zeta_{rec}(\bar{\mathbf{x}})$. Moreover, as reported in Section IV-C, for an input FPR of 5%, detectors based on all three criteria show similar detection performance, with the critic-based detector (D_{cr}) outperforming the rest when the input FPR is reduced to 0.1%.

IV. EXPERIMENTAL EVALUATION

In this section, we conduct a comprehensive performance analysis of the proposed detection methods using the publicly available TEXBAT dataset [22], [23]. Our evaluation begins with binary classification analysis under classical supervised learning and leave-one-out training strategy, which highlights the limitations of models trained under supervised learning. Subsequently, we present the results obtained from our proposed detection models, designed to address the limitations effectively. Lastly, we compare the performance of our proposed models with statistical hypothesis-based spoof detection methods, highlighting the superior performance of our approach.

A. TEXBAT DATASET

The TEXBAT dataset, provided by the RadioNavigation Laboratory (RNL) at the University of Texas in Austin, comprises

TABLE 2. Summary of Texas Spoofing Test Battery (TEXBAT) scenarios.

Abrv.	Scenario Info	Power Adv. (dB)	Spoofing Type	Synchronization	Duration (sec)
DS-0	Static Genuine	N/A	N/A	N/A	50-300
DS-1	Dynamic Genuine	N/A	N/A	N/A	50-300
DS-2	Static Over-Powered	10	Time Push	Code Phase Prop.	50-300
DS-3	Static Matched-Power	1.3	Time Push	Freq. Lock Mode	50-300
DS-4	Static Matched-Power	0.4	Position Push	Freq. Lock Mode	100-350
DS-5	Dynamic Over-Powered	9.9	Time Push	Code Phase Prop.	50-300
DS-6	Dynamic Matched-Power	0.8	Position Push	Freq. Lock Mode	50-350
DS-7	Static Matched-Power	Matched	Time Push	Carrier Aligned	100-450

high-fidelity binary recordings representing diverse spoofing scenarios involving civilian GPS L1 C/A signals [22], [23]. These recordings are centered at a carrier frequency of 1575.42 MHz and possess a 20 MHz bandwidth. Each sample is encoded with 16 bits and has a complex sampling rate of 25 Msps. TEXBAT stands out as one of the few publicly available datasets for evaluating anti-spoofing GPS receiver systems. For a succinct overview of the various spoofing attack scenarios, please refer to Table 2.

In Table 2, the term *code phase prop.* signifies that the spoofed signal's carrier phase changes proportionally to the code phase. *Frequency lock mode* denotes scenarios where the initial phase offset between the spoofed and authentic signals remains constant throughout the spoofing episode. *Carrier aligned* implies precise alignment of the spoofed signal with the carrier phase of the authentic signal. *Matched* suggests that the power of the spoofed signal and genuine are matched, though exact values are unspecified. Lastly, *Spoofing type* refers to deviations induced by the spoofer in time or position information.

This study uses all datasets listed in Table 2, covering both static and dynamic scenarios. The DS-0 dataset contains genuine GPS signals recorded with an antenna placed atop the RNL lab, whereas, DS-1 contains genuine signals recorded using a vehicle-mounted antenna travelling within Austin, Texas. Rest of the datasets can be divided into three spoofing attack scenarios, with varying degree of implementation difficulty as:

- **Simple:** DS-2 and DS-5 were generated using the genuine static and dynamic datasets, whereby, the spoofer enjoyed a significant power advantage over the genuine signals, and with *code phase proportional* synchronization.
- **Intermediate:** DS-3, DS-4, and DS-6 are the datasets where the spoofed signals has relatively small power advantage (essentially matched) and generated by performing a *frequency lock* with the respective genuine signals.
- **Sophisticated:** The DS-7 dataset contains the most challenging attack setting, where the spoofer's power is matched with the genuine signal, while maintaining precise carrier phase alignment with the genuine signal as well, which in turn causes low distortion in the correlation function as shown in Fig. 6.

To minimize redundancy, we concentrated our attention on attack scenarios within the time window just preceding the insertion of the spoofed signal and following the completion of correlator peak capture. This approach allowed us to thoroughly analyze the entire process of the attacker pulling off the correlation peak.

1) FEATURE EXTRACTION

We employed FGI-GSRx [33], an open-source GNSS software receiver designed for single-antenna GNSS processing, to process the raw RF samples from the TEXBAT datasets. This receiver is implemented in MATLAB, and thus, the initial feature extraction was conducted using MATLAB. The subsequent analyses presented in this study utilized seven features extracted from the satellite with the highest received power, sampled at an output rate of 50 Hz (averaging samples in a 20 ms time window). For visual inspection, the resulting features for datasets DS-4, DS-6, and DS-7 are displayed in Fig. 6. ML model training and performance analysis were conducted in Python v3.9, utilizing the Scikit-Learn [44] and PyTorch [45] libraries. To facilitate effective model learning, we scaled the training dataset features to the range of [0, 1]. In summary, our dataset comprised 65% spoofed samples and 35% genuine samples, totaling 107,500 samples. Although the total number of samples may appear as if the dataset is unbalanced, this is not strictly the case as we have ensured each scenario, whether it is genuine or attacking, contains similar total sample number. For instance, referring to Table 2, DS-0 to DS-5 contain 250 seconds of samples each, DS-6 includes 300 seconds of data, and DS-7 comprises 350 seconds of data.

2) PERFORMANCE METRICS

The performance metrics selected for our analysis are classification-based, including Accuracy (ACC), True Positive Rate (TPR), which is also known as the detection rate, and False Positive Rate (FPR), often referred to as the false alarm rate. The True Negative Rate (TNR) and Miss Rate

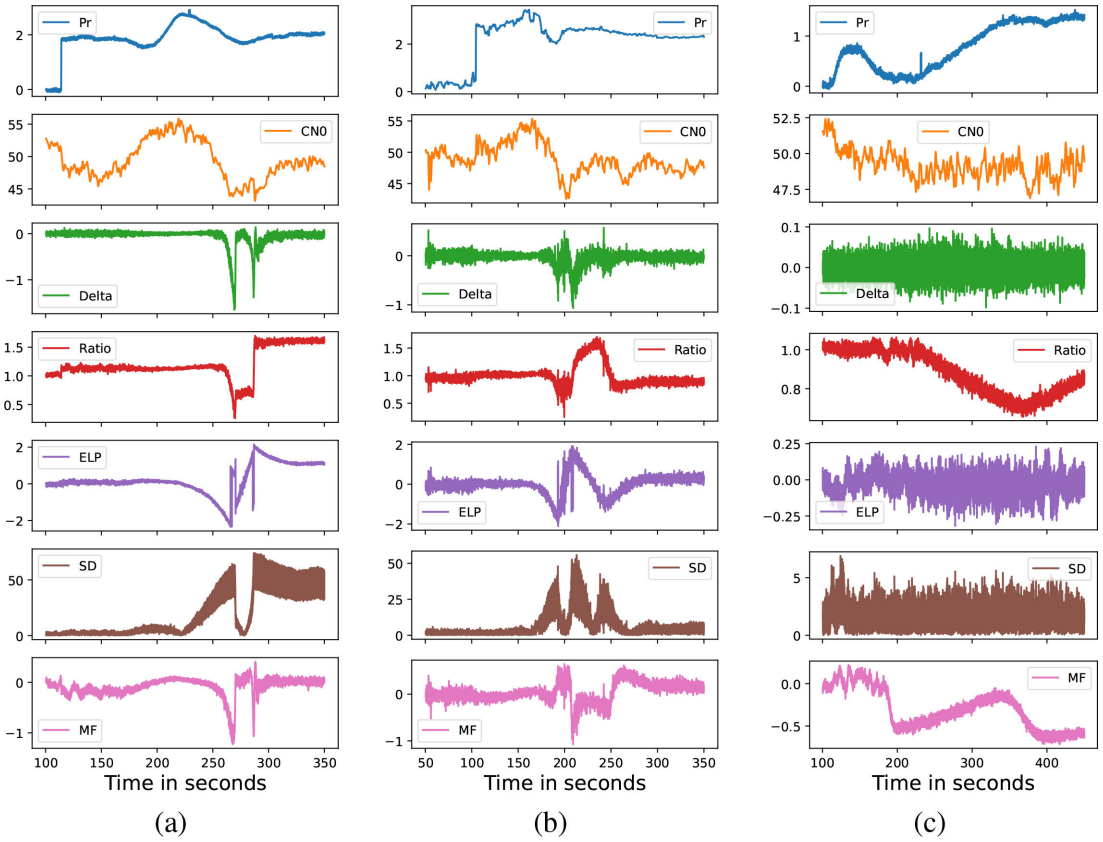


FIGURE 6. **Features extracted from a) DS-4 (PRN-23), b) DS-6 (PRN-22), and c) DS-7 (PRN-23) datasets.**

(MR) can be deduced from the FPR and TPR, respectively, and are therefore not explicitly included in the performance scores. Furthermore, for results presented in Section IV-B1, we also report the F1-score, precision, and recall results as the training dataset in this case is unbalanced. To assess the performance of the three proposed detectors, Receiver Operating Characteristic (ROC) curves are utilized as well. These curves help evaluate and compare the detectors' performance across various thresholds. In our evaluations, we consider the positive case to indicate the presence of a spoofer, while the negative case represents a genuine sample.

B. BINARY CLASSIFICATION

To evaluate the effectiveness of the extracted features in detecting spoofed signals, we employed four widely used ML classification models: the Random Forest Classifier (RFC), Gradient Boost (GB), K-Nearest Neighbours (K-NN), the Support Vector Machine (SVM), and a Fully Connected Neural Network (FCNN). We implemented RFC, GB, K-NN, and SVM using the scikit-learn [44] package, while the FCNN was developed using PyTorch [45]. We fine-tuned each model for optimal performance, configuring the RFC with 8 estimators, GB with 20 estimators, setting $K = 8$ for K-NN, and using the radial basis function (RBF) kernel for the SVM. The FCNN architecture comprised three layers with node configurations of $20 - 10 - 1$. It utilized the Parametric Rectified Linear Unit (PReLU) activation function for the first two layers and the Sigmoid function for the final output layer. Training involved minimizing the binary cross-entropy loss across 20 epochs, employing the Adam optimizer with a learning rate set at $1e^{-3}$ and a batch size of 256.

1) PERFORMANCE UNDER SUPERVISED LEARNING

In this section, we present the performance assessment of models trained within a supervised learning framework. The entire dataset was evenly divided into training and testing sets (50-50 split), and the ML and FCNN models mentioned earlier were trained on the training set. Subsequently, their performance was compared using the testing set. To ensure robustness in the evaluation, we repeated the performance assessment 10 times with different train-test splits, presenting the average classification results in Table 3. In addition to the five models trained on our featureset, we included three other models; FCNN-S [26], FCNN-B [25], and Nu-SVM [9]–for comparative analysis. These models were trained using the featuresets introduced in their respective papers. To maintain a consistent complexity level among the FCNN models, we preserved the same model architecture for FCNN-B and FCNN-S as discussed earlier.

TABLE 3. Binary classification results on the test set.

Methods	Accuracy	F1	Precision	Recall/TPR	FPR
RFC	99.45	99.69	**99.75**	99.53	**0.51**
GB	98.68	98.91	98.86	98.89	1.92
SVM	**99.65**	**99.63**	99.49	**99.68**	0.59
K-NN	98.58	98.87	99.35	98.58	1.15
FCNN	98.89	99.15	99.38	98.86	1.05
Nu-SVM [9]	91.97	93.52	98.51	88.84	3.21
FCNN-S [26]	98.32	98.65	98.67	98.61	1.64
FCNN-B [25]	92.27	93.83	97.28	90.84	4.96

Upon reviewing the scores in Table 3, it is evident that all five models trained on our featureset consistently performed well, exhibiting a minimal TPR of approximately 98%, while the SVM model achieved the highest TPR. These results underscore the efficacy and informativeness of the proposed featureset. Moreover, the scores reported by models trained on alternative featuresets also demonstrate commendable performance. However, upon closer inspection of the train-test sets, we discovered that the training set contained samples that closely resembled those in the test set. This similarity arises because the data is sampled from a time series signal, leading to a high degree of sample correlation. As a result, we emphasize caution when interpreting performance analysis based on a simple train-test split, particularly when working with high sampling rate time series data like that of GPS spoofing datasets.

2) LEAVE-ONE-OUT STRATEGY

To establish a robust baseline for our upcoming analysis, we evaluated the generalization capabilities of the top performing ML models (SVM, RFC, and FCNN) using the Leave-One-Out (LOO) strategy. In this strategy, we employed all but one dataset for training the models, reserving the remaining dataset for testing. The LOO strategy offers a distinct advantage as it evaluates the models' performance on a dataset excluded from the training set. Consequently, the model can no longer rely solely on memorization of the training samples; instead, it showcases its generalizable abilities that are effective across diverse attack scenarios. By employing this strategy on all the datasets listed in Table 2, we conducted 10 independent runs and computed the average results, which are presented in Table 4. This rigorous evaluation approach ensures that the models' performance is not influenced by the specific characteristics of a single dataset and provides a more comprehensive assessment of their generalization capabilities.

Analyzing the accuracy metrics presented in Table 4, we observe the remarkable performance of all three classifiers across a spectrum of scenarios, ranging from genuine to various spoof attack scenarios (DS-0 to DS-6). In each case, the classifiers consistently achieved accuracy surpassing 98%, with the lowest accuracy recorded at 97.36% for DS-2 when using FCNN, a still commendable result. Likewise, when examining the TPR for the attack datasets (DS-2 to DS-6), the lowest value is reported by SVM at 98.10%. These results demonstrate the robustness and reliability of the featureset and these classifiers in distinguishing between genuine and spoofed signals across different attack scenarios. However, it is essential to exercise caution when interpreting these results, considering the similarities among certain attack scenarios. Notably, DS-2 and DS-5 share common characteristics, such as a substantial relative power advantage and alignment of the spoofed signal with the code phase of the authentic signal. Similarly, DS-3, DS-4, and DS-6 exhibit comparable attack profiles, characterized by a low relative power advantage and spoofed signals operating in frequency-locked mode. Given these similarities, it is plausible that the presence of DS-3 and DS-4 in the training dataset could influence the effective classification of unseen DS-6 during testing, even if DS-6 was not included in the training data. These observations underscore the importance of carefully considering dataset characteristics and their potential impact on model performance.

Nonetheless, the DS-7 attack dataset stands apart from the others due to its unique characteristics. Spoofed signal in DS-7 attack boasts matched power with the genuine GPS signal and achieves full carrier phase alignment, making it a more subtle attack. As a result, this attack leads to minimal variations across different SQM metrics, as illustrated in Fig. 6. Specifically, the Delta, ELP, and SD metrics exhibited negligible changes in their sample statistics throughout the entire attack. Consequently, when evaluated against DS-7, all three models exhibited notably lower performance levels. FCNN, for instance, achieved the highest TPR at 44.11%. This disparity in performance underscores a fundamental challenge inherent in supervised learning, i.e., the models trained on a specific dataset tend to falter when subjected to significantly different datasets during the testing phase. The distinctive attributes of DS-7, such as its power matching and precise carrier phase alignment, made it close enough to the genuine samples such that all three classifiers classified most of DS-7 as genuine.

To further investigate the performance of SVM and FCNN models against the DS-7 dataset, we conducted an additional experiment, whereby our models were adjusted to produce logits rather than binary outcomes and were trained on all datasets except DS-7. Utilizing the DS-0 (genuine static) dataset, we computed thresholds across various FPR values using both models. Subsequently, these thresholds were used to classify DS-7 samples as genuine or spoofed, and the results are outlined in Table 5. Surprisingly, both models demonstrated significant improvements: SVM's TPR rose from 25.5% to 72.32%, while FCNN's TPR increased from 44.11% to 78.54% at an FPR of 5%. Nevertheless, these enhancements are still considerably lower than their typical performance against other attack datasets, as presented in Table 4. These outcomes underscore the necessity for

TABLE 4. Performance metrics of ML models under leave-one-out testing scenario for the TEXBAT datasets. The Metrics Are Accuracy (ACC), True Positive Rate (TPR), and False Positive Rate (FPR).

Models →	RFC				SVM				FCNN			
Test DS ↓	ACC	TPR	FPR	MR	ACC	TPR	FPR	MR	ACC	TPR	FPR	MR
DS-0	99.95	-	0.05	-	100.00	-	0.01	-	99.98	-	0.02	-
DS-1	98.73	-	1.27	-	99.70	-	0.28	-	98.80	-	1.20	-
DS-2	98.76	99.93	4.97	0.07	99.90	99.90	0.10	0.07	97.36	100.00	11.08	0.00
DS-3	99.89	100.00	0.40	0.00	99.90	100.00	0.40	0.00	99.89	100.00	0.40	0.00
DS-4	100.00	100.00	0.00	0.00	98.20	98.10	0.00	1.86	100.00	100.00	0.00	0.00
DS-5	98.25	99.99	8.41	0.01	99.00	100.00	4.59	0.00	98.19	100.00	8.72	0.00
DS-6	99.23	99.98	4.18	0.02	98.60	100.00	7.63	0.00	98.25	100.00	9.70	0.00
DS-7	36.71	33.96	0.00	66.04	28.60	25.50	0.00	74.50	46.44	44.11	0.00	55.89
Mean	91.44	88.97	2.41	11.03	90.49	87.25	1.63	12.74	92.36	90.69	3.89	9.31

TABLE 5. TPR over multiple FPRs for DS-7 dataset.

FPR	SVM	FCNN
5.00%	72.32	78.54
1.00%	67.78	73.79
0.10%	62.83	68.9

classifiers capable of detecting even the slightest feature distortions, especially in sophisticated attacks like DS-7.

C. EVALUATION OF THE PROPOSED ZERO-DAY DETECTOR

As demonstrated in the previous section, supervised training models can encounter challenges when faced with test datasets that substantially differ from the training data. To tackle the issue outlined in the preceding subsection, we employ representation learning based methods. In this framework, the model acquires the ability to discern one class from the training dataset and subsequently applies this knowledge to classify test samples, even if it has not been explicitly trained on them. This approach holds the potential to enhance the model's accuracy when dealing with novel and previously unseen datasets, making it particularly advantageous for GPS spoofing detection where attack scenarios may vary widely.

1) IMPLEMENTATION DETAILS OF THE PROPOSED MODEL

Our proposed model, illustrated in Fig. 3, comprises three FCNNs, each consisting of four layers. The input to the encoder model is a vector of size $\mathbb{R}^9$, encompassing $\mathbf{x} \in \mathbb{R}^7$ features and $\mathbf{c} \in \mathbb{R}^2$, a one-hot-encoded label representing the receiver's status (static or dynamic). The model architecture itself follows a node configuration of $20 - 10 - 5 - 4$, utilizing PReLU activation functions at its inner nodes and no activation function at the final output layer. The encoder serves to model a 2D latent distribution, characterized by a 2D mean and 2D variance vector at its output. On the other hand, the input to the decoder model is a vector of size $\mathbb{R}^4$, consisting of $\mathbf{z} \in \mathbb{R}^2$, the latent variable, and $\mathbf{c} \in \mathbb{R}^2$. The decoder model consists of four layers with a node configuration of $5-10-20-7$ and employs similar activation functions as the encoder. Finally, the critic model takes an input of $\mathbb{R}^7$, which can be either a real data sample or a fake one generated by the decoder. Its inner node architecture adopts a configuration of $20 - 10 - 5 - 1$ with PReLU activations in hidden nodes and a no activation function at the output node.

The model was exclusively trained using genuine datasets, specifically DS-0 (static) and DS-1 (dynamic), as described in Sections III-B, utilizing Algorithm 1. The model was trained over 150 epochs, with a batch size of 256. Adam optimizer was employed to update the model parameters. The learning rates for the model were progressively updated after every 50 epochs, following the sequence $[1e^{-4}, 5e^{-5}, 1e^{-5}]$. Furthermore, a training ratio of $\kappa = 5$ was applied to balance the training of the critic and encoder-decoder models. These parameter selections emerged after comprehensive hyperparameter tuning, yielding the optimal performance.

2) DETECTION SCORES UNDER MULTIPLE INPUT FPRs

Using the trained model, we devised three strategies for detecting out-of-distribution samples, as outlined in Sections III-C: D_{ℓ_2} (utilizing the ℓ_2 criterion), D_{rec} (based on reconstruction error), and D_{cr} (using critic scores), employing the statistics as given in (22), respectively. We calculated the respective thresholds ρ for each detector using the input FPR [5.0%, 1.0%, 0.1%], utilizing the trained datasets DS-0 and DS-1. With these thresholds in hand, we evaluated the performance of each detector individually using each dataset, ranging from DS-0 to DS-7, for spoofing detection. The detection performance under each detector is presented in Table 6 for D_{ℓ_2}, Table 7 for D_{rec}, and Table 8 for D_{cr}. Additionally, for visualization purposes, test statistics for

TABLE 6. Detection performance for D_{ℓ_2} based on ℓ_2 criterion. Input FPR is computed using the genuine data samples only.

Input FPR →	5.00%			1.00%			0.10%		
Test DS ↓	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
DS-0	99.35	-	0.65	99.99	-	0.01	100.00	-	0.00
DS-1	90.42	-	9.58	97.93	-	2.07	99.79	-	0.21
DS-2	95.68	100.00	18.09	99.94	99.93	0.00	99.94	99.93	0.00
DS-3	99.41	99.57	1.01	99.05	98.84	0.40	97.82	97.12	0.37
DS-4	99.17	99.14	0.29	98.43	98.34	0.00	97.19	97.03	0.00
DS-5	97.00	100.00	14.47	99.64	100.00	1.74	99.86	100.00	0.66
DS-6	96.09	98.79	16.14	93.82	93.10	2.92	82.39	78.72	0.93
DS-7	94.86	94.69	1.37	26.84	23.66	0.00	4.36	0.20	0.00
Mean	96.50	98.70	7.70	89.46	85.65	0.89	85.17	78.83	0.27

TABLE 7. Detection performance for D_{rec} based on reconstruction errors.

Input FPR →	5.00%			1.00%			0.10%		
Test DS ↓	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
DS-0	97.91	-	2.09	99.83	-	0.17	99.98	-	0.02
DS-1	90.73	-	9.27	98.10	-	1.90	99.93	-	0.07
DS-2	98.64	99.95	5.54	99.80	99.93	0.60	99.94	99.93	0.03
DS-3	99.01	100.00	3.54	99.77	100.00	0.83	99.83	99.92	0.40
DS-4	99.89	100.00	2.06	99.98	100.00	0.44	99.56	99.53	0.00
DS-5	95.66	100.00	20.92	99.27	100.00	3.51	99.98	100.00	0.12
DS-6	96.15	100.00	21.36	99.21	100.00	4.41	99.96	100.00	0.22
DS-7	82.41	81.91	6.04	62.53	60.95	0.96	40.84	38.27	0.00
Mean	95.05	96.98	8.85	94.81	93.48	1.60	92.50	89.61	0.11

each strategy are shown in Figs. 7, 8, and 9 corresponding to the ℓ_2 criterion, reconstruction error, and critic scores, respectively.

In comparison to the baseline performance of the ML algorithms, as shown in Table 4, the performance of all three proposed detectors remains similar for the attack datasets of DS-2 to DS-6 across all input FPRs. Given that all three ML models reported an FPR of approximately 1% for DS-0 and DS-1, let's focus on the case with a 1% input FPR for detailed inspection, comparing the performance of the proposed detectors for the attack datasets DS-2 to DS-6. In terms of accuracy, the proposed detector D_{rec} takes the lead, consistently maintaining an accuracy above 99%, with SVM being a close second. Regarding TPR, FCNN impressively maintains a 100% TPR consistently. The proposed detectors D_{cr} and D_{rec}, along with the RFC model, come in a close second as their TPR scores consistently stay above 99.9%. Based on these comparisons, for the DS-2 to DS-6 spoof attack datasets, it can be concluded that the proposed detection models consistently met the performance of the baseline ML models and, in some cases, even exceeded them, all without having seen any of the attack samples during their training.

For the subtle spoofing attack case of DS-7, FCNN reports the highest TPR of 78%. In contrast, the proposed detectors take a decisive lead, reporting a lowest TPR of 81.91% by D_{rec} and the highest of 94.69% by D_{ℓ_2} under the 5% input FPR. This demonstrates that for novel and subtle unseen attacks on GNSS receivers, the proposed detectors have a high chance of success compared to traditionally trained models. When comparing the proposed detectors among themselves and reducing the input FPR from 5% to 1% and then to 0.1% for DS-2 to DS-6, we observe that the accuracy and TPR reported by D_{rec} and D_{cr} consistently exceed 99.5% and 99.9%, respectively. Similarly, the FPRs consistently decrease for all three detectors as well. However, when examining the scores reported for the DS-7 dataset, with the reduction in input FPR, the TPR reported by D_{ℓ_2} sees a steep decline, dropping from 94.69% to 23.66% and then to 0.2%. This is because of the low class separation

TABLE 8. Detection performance for D_{cr} based on critic scores.

Input FPR →	5.00%			1.00%			0.10%		
Test DS ↓	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
DS-0	99.97	-	0.03	99.99	-	0.01	100.00	-	0.00
DS-1	90.98	-	9.02	98.12	-	1.88	99.82	-	0.18
DS-2	98.99	99.95	4.06	99.89	99.93	0.23	99.94	99.93	0.00
DS-3	99.88	100.00	0.43	99.89	100.00	0.40	99.89	100.00	0.40
DS-4	100.00	100.00	0.00	100.00	100.00	0.00	100.00	100.00	0.00
DS-5	91.98	100.00	38.63	96.58	100.00	16.48	99.63	100.00	1.78
DS-6	94.42	100.00	30.95	97.46	100.00	14.11	99.37	100.00	3.48
DS-7	93.64	93.36	0.00	91.04	90.65	0.00	78.13	76.74	0.00
Mean	96.23	98.88	10.39	97.87	98.43	4.14	97.10	96.11	0.73

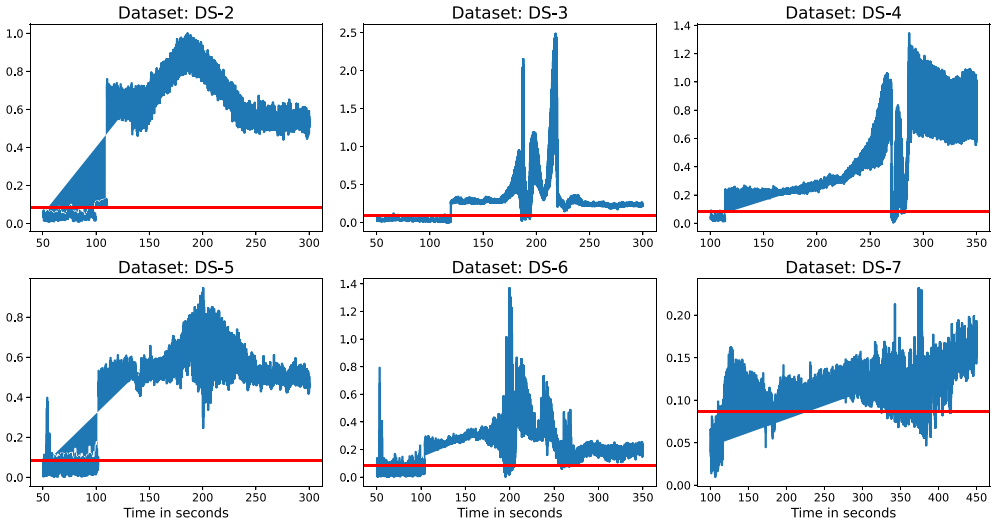


FIGURE 7. Test statistics based on ℓ_2 criterion under 5% input FPR.

between ℓ_2 metric for DS-7, as seen in Fig. 7. D_{rec} exhibits a similar downward trend in TPR, with its scores decreasing from 81.91% to 38.27%. Meanwhile, D_{cr} reports the smallest decline, decreasing from 93.36% to 76.74%. Looking at the mean results reported in Tables 4, 6, 7, and 8, we find that the proposed detector D_{cr} based on critic scores consistently maintains an accuracy and TPR above 96%, outperforming its competition.

Instead of using the input FPR for threshold detection, if we employ the FPR from the test dataset itself for threshold computation, the Figs. 7, 8, and 9 clearly reveal that the test statistic separation under D_{cr} is superior compared to D_{ℓ_2} and D_{rec}. Consequently, under this condition, it should also outperform the other detectors. To illustrate this, we present the ROC curves for all datasets under the three detectors in Fig. 10. The ROC curves indicate that the performance of D_{cr} and D_{ℓ_2} is similar, with D_{rec} being close but exhibiting the lowest reported AUC for DS-7. These results underscore the utility of representation-based learners for GPS spoof detection. We only need to train our models using high-quality genuine GPS signals, and the resulting model proves robust enough to identify novel, unseen, and sophisticated attacks.

TABLE 9. Detection performance for D_{cr} for input FPR of 1%.

	DS-5		DS-6		DS-7	
	TPR	FPR	TPR	FPR	TPR	FPR
GAN	53.75	44.24	35.77	41.83	21.98	24.13
VAE-GAN	58.62	40.56	42.95	31.74	45.86	22.52
VAE-WGAN	100	62.45	100	68.7	91.58	4.65
CVAE-WGAN	100	16.58	100	14.13	90.71	0.00

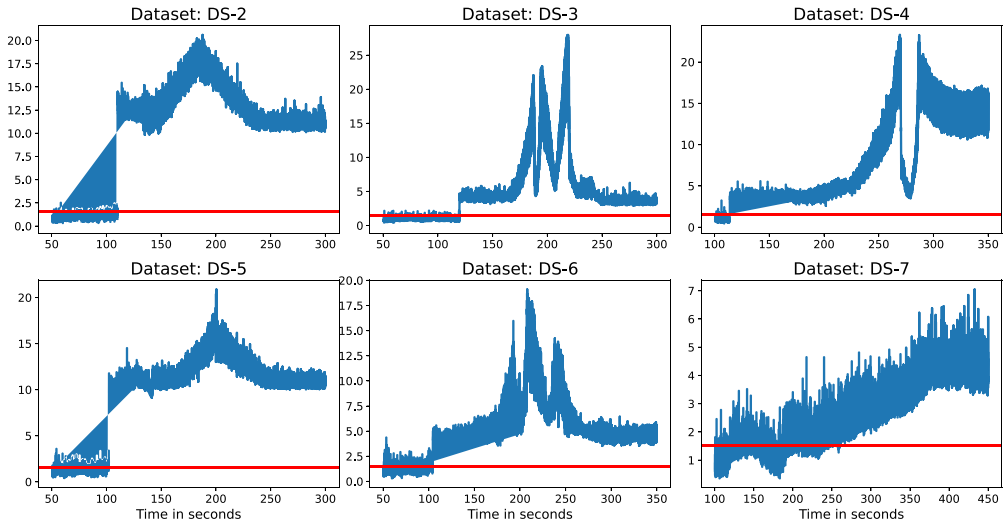


FIGURE 8. **Test statistics based on reconstruction error under 5% input FPR.**

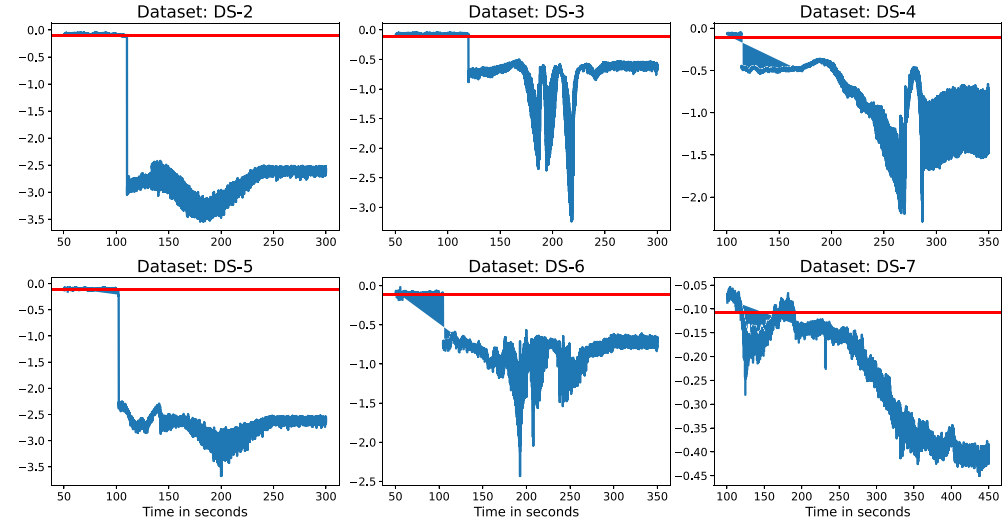


FIGURE 9. **Test statistics based on critic scores under 5% input FPR.**

3) AN ABLATION STUDY

In this section, we present an ablation study to assess the performance gains resulting from different model architectures and training strategies. The study involved four models: the vanilla GAN, VAE-GAN (GAN with an additional encoder), VAE-WGAN (WGAN-GP substituted for the vanilla GAN), and CVAE-WGAN (conditioned Encoder and Decoder models of VAE-WGAN on class labels). These models were evaluated using DS-5, DS-6, and DS-7 datasets, encompassing dynamic datasets and sophisticated attack scenarios. We focused on TPR and FPR metrics generated by D_{cr} for an input FPR of 1%, as presented in Table 9 for analysis.

The results indicate that both GAN and VAE-GAN models performed poorly. Despite various attempts with different hyperparameters, maintaining a balance between discriminator and generator loss posed a significant challenge. The discriminator excelled in distinguishing real and fake samples, leading to the generator's failure and subsequent poor performance. Introducing the WGAN-GP training strategy notably improved TPR, achieving 100% for DS-5 and DS-6, and 91% for DS-7. However, this improvement was accompanied by unreasonably high FPRs for DS-5 and DS-6, exceeding 60%. Conversely, DS-7 exhibited a low FPR of 4.65%. An analysis of latent mappings revealed that the encoder was having difficulties in mapping genuine dynamic samples close to the latent space origin, which in turn affected decoder, and consequently affecting the discriminator's ability to distinguish genuine dynamic samples from fake ones, leading to elevated FPRs for the dynamic samples.

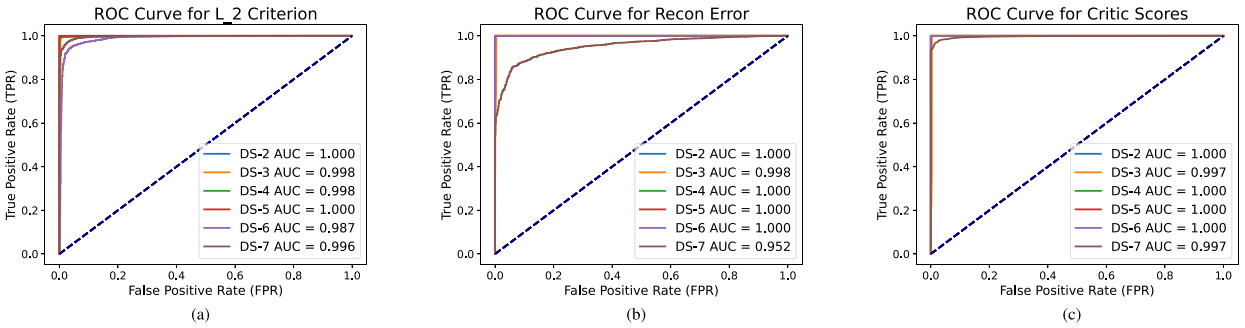


FIGURE 10. **Receiver operating characteristic (ROC) curves for a) D_{ℓ_2}, b) D_{rec}, and c) D_{cr} detectors.**

TABLE 10. **Performance comparison on TEXBAT scenario DS-7.**

Input FPR →	5 %			1 %			0.10 %		
Methods ↓	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR
Wesson *et al.* [12]	71.25	73.96	8.56	70.76	69.4	5.08	70.18	67.73	1.98
Manfredini *et al.* [21]	69.35	68.34	11.68	63.22	65.87	7.67	59.68	62.18	3.58
Sun *et al.* [17]	56.34	57.68	8.64	53.84	54.08	5.68	51.35	52.54	1.28
Zhou *et al.* [19]	78.55	81.65	9.32	79.89	79.35	6.58	77.76	75.97	2.85
Proposed D_{ℓ_2}	**94.86**	**94.69**	1.37	26.84	23.66	**0.00**	4.36	0.20	**0.00**
Proposed D_{rec}	82.41	81.91	6.04	62.53	60.95	0.96	40.84	38.27	**0.00**
Proposed D_{cr}	93.64	93.36	**0.00**	**91.04**	**90.65**	**0.00**	**78.13**	**76.74**	**0.00**

Addressing this issue, we conditioned the encoder and decoder models on class labels for static and dynamic classes in the CVAE-WGAN model. This modification allowed the encoder to smoothly map genuine training samples from both static and dynamic datasets around the latent space origin. As a result, FPR scores for DS-5 and DS-6 dropped below 20%, maintaining 100% TPR. DS-7 experienced a slight decline in TPR but achieved a 0% FPR. This enhancement in model performance demonstrates the importance of conditioning the models on class labels to mitigate false positive rates in dynamic datasets, ensuring robust spoof detection across various attack scenarios.

4) COMPARISON WITH RECENT WORKS

In this section, we undertake a comparative analysis of the proposed spoof detectors with statistical hypothesis-based spoof detection methods, as outlined in previous studies [12], [17], [19], [21]. The basis for this comparison lies in the fact that these methods also leverage features readily available at the tracking stage of a standard GPS receiver, including received power, C/N_0, and multiple correlation finger outputs. For the purpose of this evaluation, we concentrate on the DS-7 scenario dataset, as it presents the most challenging scenario among all the TEXBAT datasets. Wesson et al. [12] employ received power, C/N_0, and the symmetric differences metric (10) in their spoof detection approach. In contrast, Manfredini et al. [21] utilize AGC values from the RF front end and the SQM metric (11), where the AGC values are the inverses of the received power, as opposed to our method. Sun et al. [17] combine ratio and delta metrics, while Zhou et al. [19] employ a custom SQM metric computed using five correlator pairs. Across all these methods, the detection window is consistently set at 2 seconds (100 samples), and the detection thresholds are computed using [5%, 1%, 0.1%] false positive rates (FPR) as specified in their respective papers. To ensure consistency in the comparison, we merge the normal and multipath hypotheses into a single non-spoof hypothesis for [12]. Similarly, we combine the spoofing and jamming hypotheses into a single spoofing hypothesis. Additionally, in the case of [17], the presented results utilize the SQM composite based on the probability of false alarm (PFA) with the weighting factor $\lambda = 0.5$, as it yielded superior performance.

Using the nomenclature adopted in Table 6, we present the performance of these methods in Table 10. Upon examining the true positive rate (TPR) across all input false positive rates (FPRs), it is evident that [17] exhibits the lowest performance. This outcome aligns with expectations as it is a transient detector, relying on conventional SQM metrics that are sensitive only during the correlator peak pull-off phase; once the pull-off exceeds 1 chip length, their detection rate rapidly declines to zero. In contrast, [19] demonstrates high sensitivity to correlation peak distortion, swiftly detecting the peak pull-off due to the utilization of multiple equi-distant correlators. Nevertheless, the TPR of [19] is approximately 10 points lower than that of the D_{cr} across input FPRs of

5% and 1%, and similar at 0.1%. On the other hand, despite employing power and SQM metrics, both [12] and [21] exhibit lackluster performance in the sophisticated attack scenario of DS-7. This can be attributed to the spoofer's minimal power advantage and their achievement of challenging frequency matching with the genuine GPS received signal. Overall, Table 10 demonstrates that the proposed detector D_{cr} not only achieves a higher detection rate but also yields lower FPRs, highlighting its effectiveness in comparison to existing methods.

V. CONCLUSION & FUTURE WORK

In this study, we addressed the critical challenge of spoof attacks on GNSS receivers, which are integral to various systems in our daily lives. Leveraging features computed at the RF and tracking stages of a typical GNSS receiver, we shed light on the effectiveness and limitations of supervised ML models. These models exhibit outstanding detection capabilities when tested on samples resembling the distribution of the training datasets. However, when faced with the subtle attack scenario of DS-7 in a leave-one-out training strategy, they are susceptible to failure, with the highest detection rate reaching only 44%, as reported by FCNN. To overcome this challenge, we introduced a representation learning-based spoof detection model comprising three FCNNs, representing a fusion of VAE and GAN architectures. The model was meticulously trained using genuine TEXBAT datasets, encompassing genuine GPS signals recorded by both static (DS-0) and dynamic (DS-1) receivers. Subsequently, we devised three distinct spoof detectors based on the three trained FCNNs. Our experimental results demonstrated that the proposed detectors matched the performance of supervised ML models for DS-2 to DS-6 datasets, achieving an impressive detection rate of approximately 99%. Furthermore, our models showcased superior detection rates for DS-7 across various input FPRs, with a highest TPR of 94.7% with an FPR of 1.37% attained by the D_{ℓ_2} detector. Notably, the D_{cr} detector, based on critic model scores, emerged as the most robust option, consistently outperforming other detectors under multiple input FPR settings. Moreover, these lightweight detectors, once trained offline, can be seamlessly integrated into GNSS receivers through a firmware update, incurring minimal computational overhead. In summary, our approach offers a promising solution to safeguard GNSS systems against increasingly sophisticated spoofing attacks, ensuring their continued reliability and security in essential applications.

An essential requirement underpinning our effective detector's performance is the availability of high-quality genuine training data encompassing diverse scenarios. This includes genuine data collected in open fields, rural low-rise building areas, and urban settings with high-rise buildings. Access to such diverse genuine data ensures the proposed model's capacity to generalize effectively across varied operating conditions. Furthermore, an intriguing avenue for future research involves treating the training samples as a time series. This approach could lead to the development of a detection model that conducts spoof detection within a time window rather than on a per-sample basis, as commonly performed in our and other recent studies.

REFERENCES

[1] J. Zidan, E. I. Adegoke, E. Kampert, S. A. Birrell, C. R. Ford, and M. D. Higgins, "GNSS vulnerabilities and existing solutions: A review of the literature," *IEEE Access*, vol. 9, pp. 153960–153976, 2021.

[2] *Systems Engineering and Integration Interface Specification is-GPS-200G*, GPS Directorate, Los Angeles, CA, USA, 2012.

[3] *European GNSS (Galileo) Open Service Signal in Space Interface Control Document*, European Union, Brussels, Belgium, 2010.

[4] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. Kintner, "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," in *Proc. 21st Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS)*, 2008, pp. 2314–2325.

[5] S. V. S. Chauhan and G. X. Gao, "Synchrophasor data under GPS spoofing: Attack detection and mitigation using residuals," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3415–3424, Jul. 2021.

[6] S. De Silva, J. Kim, E. Cotilla-Sanchez, and T. Hagan, "On PMU data integrity under GPS spoofing attacks: A sparse error correction framework," *IEEE Trans. Power Syst.*, vol. 36, no. 6, pp. 5317–5332, Nov. 2021.

[7] R. Huang and Y. Li, "False Phasor data detection under time Synchronization attacks: A neural network approach," *IEEE Trans. Smart Grid*, vol. 13, no. 6, pp. 4828–4836, Nov. 2022.

[8] Y. Dang, C. Benzaid, B. Yang, T. Taleb, and Y. Shen, "Deep-ensemble-learning-based GPS spoofing detection for cellular-connected UAVs," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25068–25085, Dec. 2022.

[9] G. Aissou, S. Benouadah, H. El Alami, and N. Kaabouch, "Instance-based supervised machine learning models for detecting GPS spoofing attacks on UAS," in *Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2022, pp. 208–214.

[10] R. Calvo-Palomino, A. Bhattacharya, G. Bovet, and D. Giustiniano, "Short: LSTM-based GNSS spoofing detection using low-cost spectrum sensors," in *Proc. IEEE 21st Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Aug. 2020, pp. 273–276.

[11] C. Kim, S.-Y. Chang, D. Lee, J. Kim, K. Park, and J. Kim, "Reliable detection of location spoofing and variation attacks," *IEEE Access*, vol. 11, pp. 10813–10825, 2023.

[12] K. D. Wesson, J. N. Gross, T. E. Humphreys, and B. L. Evans, "GNSS signal authentication via power and distortion monitoring," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 2, pp. 739–754, Apr. 2018.

[13] X. Wei, M. N. Aman, and B. Sikdar, "Exploiting correlation among GPS signals to detect GPS spoofing in power grids," *IEEE Trans. Ind. Appl.*, vol. 58, no. 1, pp. 697–708, Jan. 2022.

[14] D. M. Akos, "Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC)," *Navigation*, vol. 59, no. 4, pp. 281–290, Oct. 2012.

[15] R. E. Phelts, *Multicorrelator Techniques for Robust Mitigation of Threats to GPS Signal Quality*. Stanford, CA, USA: Stanford Univ., 2001.

[16] O. M. Mubarak and A. G. Dempster, "Performance comparison of ELP and DELP for multipath detection," in *Proc. 22nd Int. Tech. Meeting Satell. Division The Inst. Navigat. (ION GNSS)*, 2009, pp. 2276–2283.

[17] C. Sun, J. W. Cheong, A. G. Dempster, H. Zhao, and W. Feng, "GNSS spoofing detection by means of signal quality monitoring (SQM) metric combinations," *IEEE Access*, vol. 6, pp. 66428–66441, 2018.

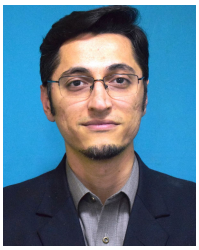
[18] A. M. Khan, N. Iqbal, A. A. Khan, M. F. Khan, and A. Ahmad, "Detection of intermediate spoofing attack on global navigation satellite system receiver through slope based metrics," *J. Navigat.*, vol. 73, no. 5, pp. 1052–1068, Sep. 2020.

[19] W. Zhou, Z. Lv, X. Deng, and Y. Ke, "A new induced GNSS spoofing detection method based on weighted second-order central moment," *IEEE Sensors J.*, vol. 22, no. 12, pp. 12064–12078, Jun. 2022.

[20] E. Schmidt, N. Gatsis, and D. Akopian, "A GPS spoofing detection and classification correlator-based technique using the LASSO," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 6, pp. 4224–4237, Dec. 2020.

[21] E. G. Manfredini, D. M. Akos, Y.-H. Chen, S. Lo, T. Walter, and P. Enge, "Effective GPS spoofing detection utilizing metrics from commercial receivers," in *Proc. Int. Tech. Meeting Inst. Navigat.*, Jan. 2018, pp. 672–689.

[22] T. E. Humphreys, J. A. Bhatti, D. Shepard, and K. Wesson, "The Texas spoofing test battery: Toward a standard for evaluating gps signal authentication techniques," in *Proc. Radionavigation Lab. Conf.*, 2012. [Online]. Available: http://hdl.handle.net/2152/63229
[23] T. Humphreys, *Texbat Data Sets 7 and 8*. Austin, TZ, USA: Univ. Texas, 2016.
[24] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.
[25] S. C. Bose, "GPS spoofing detection by neural network machine learning," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 37, no. 6, pp. 18–31, Jun. 2022.
[26] 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," *J. Navig.*, vol. 71, no. 1, pp. 169–188, 2018.
[27] 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 Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–6.
[28] A. Iqbal, M. N. Aman, and B. Sikdar, "Machine learning based time synchronization attack detection for synchrophasors," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2023, pp. 1–6.
[29] P. Borhani-Darian, H. Li, P. Wu, and P. Closas, "Deep neural network approach to detect GNSS spoofing attacks," in *Proc. 33rd Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS)*, Monterey, CA, USA, Oct. 2020, pp. 3241–3252.
[30] J. Li, X. Zhu, M. Ouyang, W. Li, Z. Chen, and Q. Fu, "GNSS spoofing jamming detection based on generative adversarial network," *IEEE Sensors J.*, vol. 21, no. 20, pp. 22823–22832, Oct. 2021.
[31] S. Dasgupta, M. Rahman, M. Islam, and M. Chowdhury, "Prediction-based GNSS spoofing attack detection for autonomous vehicles," 2020, *arXiv:2010.11722*.
[32] *GPS TTFF and Startup Modes*. Accessed: Dec. 2, 2023. [Online]. Available: https://www.measurementsystems.co.uk/docs/ttffstartup.pdf
[33] K. Borre, I. Fernández-Hernández, J. A. López-Salcedo, and M. Z. H. Bhuiyan, *GNSS Software Receivers*. Cambridge, U.K.: Cambridge Univ. Press, 2022.
[34] A. J. Jahromi, A. Broumandan, S. Daneshmand, G. Lachapelle, and R. T. Ioannides, "Galileo signal authenticity verification using signal quality monitoring methods," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, Jun. 2016, pp. 1–8.
[35] W. Wang, N. Li, R. Wu, and P. Closas, "Detection of induced GNSS spoofing using S-curve-bias," *Sensors*, vol. 19, no. 4, p. 922, Feb. 2019.
[36] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*. New York, NY, USA: Academic, 2015.
[37] A. Iqbal, M. Naveed Aman, and B. Sikdar, "Representation learning based time synchronization attack detection for synchrophasors," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Nov. 2023, pp. 1–6.
[38] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen, "Decoupling representation learning and classification for GNN-based anomaly detection," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1239–1248.
[39] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[40] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[41] A. Iqbal and A.-K. Seghouane, "An α-divergence-based approach for robust dictionary learning," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5729–5739, Nov. 2019.
[42] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–43, Feb. 2019.
[43] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–7.
[44] F. Pedregosa, S. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Dec. 2011.
[45] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, Dec. 2019, pp. 8024–8035.



ASIF IQBAL (Member, IEEE) received the B.S. degree in telecommunication engineering from the National University of Computer and Emerging Sciences (NUCES)-FAST, Peshawar, Pakistan, in 2008, the M.S. degree in wireless communications from LTH, Lunds University, Sweden, in 2011, and the Ph.D. degree in electrical and electronics engineering, The University of Melbourne, Melbourne, Australia, in 2019. He is currently a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Previously, he was on the Faculty of NUCES, as an Assistant Professor. His research interests include signal processing, deep learning, sparse signal representations, and privacy preserving machine learning.

MUHAMMAD NAVEED AMAN (Senior Member, IEEE) received the B.Sc. degree in computer systems engineering from UET Peshawar, Khyber Pakhtunkhwa, Pakistan, in 2006, the M.Sc. degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, in 2008, and the M.Eng. degree in industrial and management engineering and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2012. He is an Assistant Professor with the University of Nebraska-Lincoln. His research interests include IoT and network security, hardware systems security and privacy, wireless and mobile networks, and stochastic modeling.

BIPLAB SIKDAR (Senior Member, IEEE) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was on the Faculty of Rensselaer Polytechnic Institute, from 2001 to 2013, first as an Assistant and then an Associate Professor. He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include wireless networks, security for IoT, and cyber physical systems.