IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 25, 2022, accepted April 12, 2022, date of publication April 22, 2022, date of current version May 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169420

GANSAT: A GAN and SATellite Constellation Fingerprint-Based Framework for GPS Spoof-Detection and Location Estimation in GPS Deprived Environment

DEBASHRI ROY[1], TATHAGATA MUKHERJEE[2], ALEC RIDEN[2], JARED PAQUET[3], EDUARDO PASILIAO[4], AND ERIK BLASCH[4], (Fellow, IEEE)

[1]Department of Computer Science, University of Central Florida, Orlando, FL 32826, USA
[2]Department of Computer Science, The University of Alabama in Huntsville, Huntsville, AL 35899, USA
[3]Department of Computer Science, University of Florida, Gainesville, FL 32611, USA
[4]Munitions Directorate, Air Force Research Laboratory, Arlington, VA 22203, USA

Corresponding author: Debashri Roy (debashri.roy.15@knights.ucf.edu)

The research was supported by the Munitions Directorate, Air Force Research Laboratory, Eglin AFB, FL 32542 USA.

ABSTRACT This paper presents a robust system for mitigating adversarial and natural GPS disruptions by presenting: (1) a software-based defense mechanism against spoofing attacks using generative adversarial networks (GANs), The system detects unauthorized or spoofed GPS signals from a hardware based spoofer, and (2) deep neural network models to infer positioning information in GPS-degraded /denied environments using the novel idea of GPS satellite constellation fingerprint. As the GAN and Satellite constellation fingerprinting are used together in a unified framework, we call it the ''GANSAT positioning system.'' Intuitively, the GANSAT neural networks implicitly learn a representation of the aggregation of the hardware fingerprints of the satellite's in the GPS constellation at a given location and time. To demonstrate the approach, raw GPS signals were collected from the satellite transmitters using a software defined radio (SDR) at five different locations in the Florida panhandle area of the United States. Additionally, a GPS spoofer is implemented using a SDR and an open source software and used in an uncontrolled laboratory environment for spoofing the GPS signals at the aforementioned locations. In our experiments, the GANSAT framework yields ∼99.5% accuracy for the task of identifying and filtering the spoofed GPS signals from real ones. It also achieves ∼100% accuracy for the task of location estimation.

INDEX TERMS GPS, GNSS, positioning, machine learning, generative adversarial nets, deep neural network.

I. INTRODUCTION

From its inception in the early 1990's the Global Positioning System (GPS) has slowly infused itself into the fabric of our daily lives. Large scale deployment of smartphones have ushered in the era of ubiquitous use of GPS. From walking, driving, buying food, getting money to finding a rideshare and national security, GPS is everywhere. Such is it's usefulness and integration into our lives that many of us assume that it is omnipresent; that there can never be a situation when GPS might not be there. However this cannot be further from the truth as in reality GPS systems are susceptible to both natural and adversarial disruption. The situation is further complicated by the fact that the conditions that might lead to such disruptions are easily created and there is no fallback system available for addressing such failures.

Regardless of whether one is using the GPS device built into a smartphone or a standalone GPS system, the underlying basic principle for obtaining positioning, navigation and timing (PNT) information is the same. GPS system uses time synchronization across satellites and receivers for computing the *time difference of arrival (TDoA)* of the signals at a receiver. A time synchronized GPS receiver computes the PNT information with the data contained in the received signal from multiple GPS satellites. The satellites visible at any given time from a given region form a sub-constellation of the

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

entire GPS constellation. The signals from each satellite in the visible constellation at any point on Earth contains information about the time at which the signal was transmitted and the position of the satellite at that instant. This information can be parsed and used by a GPS receiver for TDoA computation. Though this method for obtaining the PNT information is simple at the core, a lot of research has gone into improving the operational accuracy of such systems. Three contemporary areas of PNT research include: GPS signals corruption, machine learning (ML) signals analysis, and satellite signal fingerprinting.

GPS uses the idea of ''spread spectrum'' on the carrier frequency to distribute the satellite transmissions over a large frequency band in order to maintain a low power profile [1]. Spread spectrum is used to utilize a larger bandwidth than that required by the original message while maintaining the same power. A spread spectrum signal does not have a clearly identifiable peak which makes it more difficult to distinguish it from the background noise. Since GPS uses spread spectrum for the transmission in the L1 (1575.42 MHz) and L2 (1227.60) bands, the signals are devoid of sharp peaks at a ground based receiver. Though this makes it is more difficult to intercept the signal by detecting the same, especially in environments with low SNR (signal-to-noise ratio), this also makes it easy to overwhelm them using a high power spoofer placed near the ground based receiver. Since the GPS satellites are far away, the signal strength at a ground-based receiver is expected to be fairly low by design and this makes them susceptible to corruption by a high power spoofer. Thus due to a combination of several factors, the GPS receiver is prone to spoofing attacks whereby fake information from a high power spoofer can be used to fool the receiver into using wrong information for positioning. This false information can be passed on to other applications thereby corrupting their functions as well. With the wide availability of software defined radios (SDRs), it has become easy to build commercial off-the-shelf (COTS) ''GPS spoofers,'' for use both in commercial and civilian settings. Such capability hastens the need to ensure the received GPS signals' reliability in order to protect against the consequences of such attacks, which can be catastrophic in many cases.

GPS spoofing attacks are common and equally likely in the communication and dissemination settings with serious consequences in both the cases. For example in the communication setting, spoofing attacks can be done to cause severe disruption in air traffic, can be used for hijacking of delivery drones and cause large scale disruption of transportation and communication networks that rely on precise positioning information from GPS. On the dissemination side, GPS spoofing attacks can be used disrupt functioning of surveillance and reconnaissance systems that rely on precise timing information for time synchronization. It can also cause large scale disruption in services that use precise timing information from GPS systems. For example disruption in dissemination of precise timing information can have huge impact on the stock market, the functioning of banking and financial sectors, the working of infrastructure services like power grid to name a few. It must be noted that there is a protected GPS band, with special a encryption system to ensure the security and authenticity of the transmitted data. This is used by organizations related to national security like the army, navy and the air force. However this band is also vulnerable since GPS disruptions are often caused by state actors who have more resources and power at their disposal to affect such attacks. The massive proliferation of GPS receivers and the ease of use of software-defined solutions for implementing spoofers, coupled the fact that the characteristics of the spoofing hardware is unknown in most cases, makes it impossible to implement a manually configurable solution for the problem of detecting spoofed GPS signals. These factors necessitate the use of robust techniques (e.g., machine learning) that are agnostic to the presence of intelligent adversaries for the problem of detecting and mitigating GPS spoofing attacks.

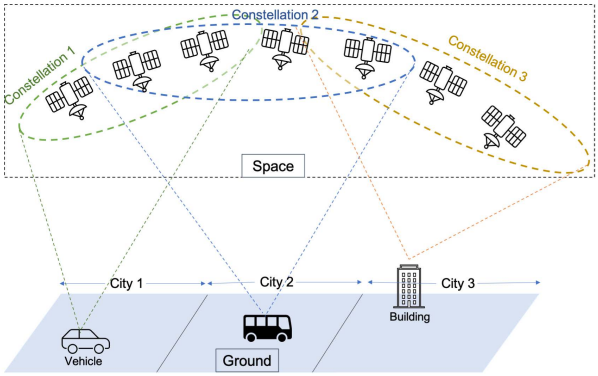


FIGURE 1. A pictorial representation of varying GPS constellations as visible from different cities.

In recent years, there has been a proliferation of autonomous systems using machine learning (ML) algorithms for inference with large scale radio frequency (RF) data. A number of ML algorithms, such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Non-parametric Bayesian Classifier and Neural Networks (NN), have already been used for various tasks in radio frequency machine learning [2]. However, deeper variants of NNs have proven to be superior than traditional ML techniques by providing a way to implement an end-to-end system for automatically computing and exploiting deep feature representations of the underlying signal data [3]. It must however be noted that ML techniques are susceptible to malicious interference since the underlying assumption of any standard ML algorithm is that the data (training and test data for supervised algorithms and the data in general for unsupervised methods), faithfully capture and represent the underlying data distribution. This assumption fails in adversarial settings where an intelligent adversary can tamper with the data and introduce flaws in the learned model. In order to circumvent the problem of adversarial interference in the operation of learning algorithms, robust learning techniques are needed that can

detect and eliminate maliciously modified (or crafted) data. One such technique is the generative adversarial network (GAN) [4], which was proposed with the goal of generating and discriminating ''real'' samples from a given distribution from ''fake'' ones generated by a generative system. A by-product of a GAN is a ''discriminator,'' which can be used to differentiate between real and fake data. Thus in a GAN, which consists of a generator and a discriminator, the goal of the generator is to generate realistic fake samples (fake data) from a given data distribution whereas that of the discriminator is to differentiate the real samples from the fake ones. This allows us to use a GAN to implement a robust learning system that can be agnostic to adversarial interference.

GAN training proceeds iteratively with the generator and discriminator being trained in tandem. The generator is trained to generate realistic fake samples using implicit feedback from the discriminator which is trained to differentiate between the real and fake samples. By the end of the training process, the discriminative NN [5] learns to differentiate between real and generated (fake or spoofed) signals obtained from the generator. It must be noted that as the GAN training proceeds, the generator learns to sample from the distribution underlying the real data, using implicit feedback from the discriminator, in order to generate realistic signals. On the other hand, the discriminator iteratively becomes better at distinguishing the real from the fake data. As a result, at the conclusion of the GAN training, the optimized discriminator is robust in the sense that it can distinguish between real and spoofed signals, regardless of the emitter being used to generate and transmit the fake data. Likewise, at the end of the GAN training, the generator learns to generate spoofed signal data that mimics the real signals. Thus in the end the discriminator learns to protect against spoofing attacks mounted by an adversary that can sample from the distribution underlying the real data. hence it can protect against intelligent adversarial attacks.

It must be pointed out that the GAN approach for spoofed data detection has been shown to outperform classical ML techniques, like SVM or the idea of outlier detection [6]. Traditional ML methods use ''expert engineered'' features, which are optimized for specific operational environments and hence may not generalize well to other scenarios. This is more so for the case of RF signals, which are very easily affected by the conditions of the surrounding environment. Hence, it becomes harder to hand-engineer expert features that are robust to changes in the operating conditions especially for the RF domain. As a result classical methods like SVM fail to perform well in such cases. Furthermore when an intelligent adversary spoofs data they do it in a way that the spoofed data is not an outlier. As a result classical outlier detection methods also fail with such data. GAN-based methods, on the other hand, provide an end-to-end solution without the need for expert engineered features. They automatically learn intrinsic representations of the signal data, which are robust to random in the operating environment. Since a GAN works by learning a representation of the underlying data distribution, adversarial spoofing methods also fail. This is because the GAN trained discriminator is able to efficiently identify the spoofed data as it is generated using a perturbed version of the data distribution [6].

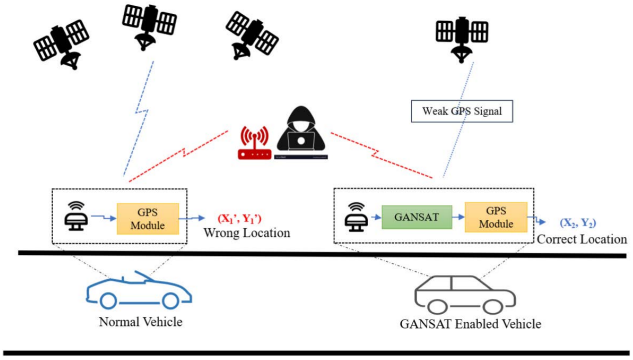


FIGURE 2. A scenario of using GANSAT to detect the spoofed GPS signals and predict correct location.

GPS is a United States' Global Navigation Satellite System (GNSS) that provides positioning, navigation and timing information to ground based receivers. The system consists of 31 satellites deployed around the Earth at an altitude of 11,000 miles (which is referred to as medium earth orbit (MEO)). Each of these satellites has a stable atomic clock, which is synchronized with that of the other satellites. The location and time of each satellite is continuously estimated from ephemeris parameters, broadcast on L1 C/A (1575.42 MHz) and L2 P/Y (1227.60MHz) band with a data rate of 50bps [7]. The government is in the process of fielding three new GPS channels designed for civilian use: L2C, L5, and L1C. The legacy civil signal, namely L1 C/A, will continue broadcasting, bringing the total number of GPS signal channels to four. The L2C is the second civilian GPS signal, designed specifically to meet commercial needs. The L5 GPS band is the third civilian GPS channel, designed to meet demanding requirements of safety-of-life transportation and other high performance applications. Its name refers to the U.S. designation for the radio frequency used by the signal which is centered at 1176 MHz. L5 is broadcast in a radio band reserved exclusively for aviation safety services. It features higher power, greater bandwidth, and an advanced signal design. Future aircraft will use L5 in combination with L1 C/A to improve accuracy (via ionospheric correction) and robustness (via signal redundancy).

A GPS receiver on earth computes its position by simultaneously calculating its distance from at least four GPS satellites, using the idea of time difference of arrivals (TDoA). Each satellite also broadcasts a unique pattern of 1,023 plus and minus signs known as a ''pseudo-random noise'' (PRN) code. The satellites actually broadcast two sets of PRN codes: one for civilian use and the other for the U.S. military use. Civilian PRN codes are unencrypted and published in a public database, while military codes are encrypted in a pattern that

is predictable only if the receiver has access to a secret key. The satellites that are visible to a GPS receiver change based on the position of the receiver on earth and the time of day at which the reception was made. The PRN codes are an important element of code division multiple access (CDMA) based satellite navigation systems. Since each satellite within the GPS constellation has a unique PRN code, that it transmits as part of the C/A navigation message, this code allows any receiver to identify the satellite(s) with which it is communicating. The PRN codes act as spreading codes in the spread-spectrum communications system, and must be carefully chosen to minimize the interference between each satellite signal. Failure to do so would leave the system open to so-called CDMA noise, potentially degrading performance to unworkable levels.

In this paper the GANSAT framework implicitly leverages the PRN codes, through learning on the received signal characteristics. This allow us to fingerprint different ''visible'' parts of the constellation from various locations on Earth at different times of day. Note that though there is only one constellation of GPS satellites (which includes all the satellites), for better clarity and understanding, GANSAT treats each ''visible part of the GPS constellation at a location at a given time'' as a different *sub-constellation* and names them accordingly (e.g., C_1, C_2) etc.). Fig. 1 illustrates the fact that different cities utilize different satellite sub-constellations at different times of the day. Hence the locations are calculated from the signals of different satellites at each location and at each given time of day.

A combination of unique PRN numbers from the various GPS satellites with their intrinsic hardware characteristics affect the collective source signal received and thus each visible constellation has an unique ''fingerprint'' of its own. We call these the ''satellite constellation fingerprint,'' ''satellite constellation signature,'' or simply ''constellation fingerprint.'' In GANSAT, a Convolutional Neural Network (CNN) based technique is used to learn these characteristics automatically in an end-to-end framework and build a ''constellation fingerprinting'' system for satellite constellation identification. To automatically extract and learn this fingerprint for a given location at a given time, we use the so called I/Q information of the received signal at the given location on earth at the given time. Any RF source signal consists of an In-phase (I) and a Quadrature-phase (Q) component. GANSAT uses this raw signal data as input to automatically ''learn'' the intrinsic signal characteristics, which can then be used as a ''fingerprint'' for the task of constellation identification [6] and hence for positioning in GPS degraded environments. A typical example of using GANSAT in moving vehicles is presented in Fig. 2. The trained-GANSAT module deployed in the vehicle helps to predict the correct location not only in a weak GPS environment, but also in the presence of spoofers.

The GANSAT framework solves two problems, which are independent in their own right: (1) first it demonstrates the use of generative adversarial learning for the task of detecting spoofed GPS signals in an uncalibrated real RF environment, and (2) second, it introduces the idea of ''GPS constellation fingerprinting'' in *GPS-denied*, *weak-GPS* as well as *strong-GPS* environments. We establish the efficacy of this framework for the task of location prediction in these environments by conducting ''real world'' experiments in the Florida Panhandle area of US. In our experiments, once the spoofed signals are eliminated using adversarially trained discriminator, a CNN estimates the location of the GPS receiver using the learned satellite constellation fingerprint. It is to be noted that *GPS-denied* and *weak-GPS* environments refer to settings where the GPS satellites are available and transmitting but due to the nature of the RF environment at the receiver, the received signal strength (RSS) is not sufficient to ''lock on'' to the minimum number of required satellites for the purpose of PNT.

The main contributions of this paper are as follows:

1) Formulating a GAN-based framework to disambiguate *spoofed GPS* signals from *real* ones and estimate the position of a receiver using the filtered real raw GPS data in a GPS-degraded environment. GANSAT has two parts: a GAN-trained GPS spoofer-detector and a position estimator that leverages ''constellation signatures''. The spoofer-detector is tested on spoofed GPS data obtained using a software-defined GPS spoofer. The position estimator is trained and tested on real GPS satellite constellation data obtained in three settings (denied, weak and strong GPS).
2) Design and implementation of a 9-layer 1-dimensional CNN (1D-CNN) used in the GANSAT framework to classify and predict the location using the signatures of the visible ''GPS constellation'' from the GAN-filtered **real-GPS** signals. This provides a robust mechanism for estimating the approximate location of the GPS receiver using the idea of satellite constellation fingerprinting. Note that in general the task of location prediction can be modeled as a regression problem. However in the case of GANSAT we model it as a classification problem as our goal is to establish that the satellite constellation signature can be used for inferring the location. Thus the CNNs we use are modeled as classifiers and not regressors.
3) Improving the 9-layer 1D-CNN to a 3-layer 2 dimensional CNN (2D-CNN) model to estimate the location using constellation fingerprints, achieving almost 100% accuracy and requiring minimal training time.
4) Designing and implementing a feature augmentation technique, which we call the ''multi-angular projection'' method. The projection is a pre-processing step used on the raw GPS received signal data that helps to bring out the spatial correlation in the signal data. This correlation is then exploited by the deep feature learners to compute discriminating features for spoofed data identification as well as position estimation.
5) Implementing a GAN based training for building a discriminator for spoofed GPS signal data. The resulting discriminator is used to differentiate real GPS signal

data from spoofed ones obtained from a hardware based spoofer. To the best of our knowledge, the proposed model is the first of its kind and provide an end-to-end robust GPS spoofer detection technique using the idea of ''satellite hardware fingerprinting.'' We must point out that this is different from the idea of ''constellation fingerprinting'' used for the task of position estimation in GPS degraded settings using the aforementioned CNN.

6) Evaluating the performance of the implemented system and establishing its efficacy using real world experiments in uncontrolled RF environments. Since the trained model is designed at the physical layer, it will not add any computational overhead in the deployment phase for practical applications. Additionally, the proposed GANSAT model is the first of its kind that provides any statistical validation of a GPS spoofer detection mechanism.

The remainder of the paper is organized as follows. Section II provides a brief background of related work. The GANSAT framework and the testbed setup are discussed in Section III and Section IV respectively. The experimental results are presented in Section V and conclusions are drawn in the last section.

II. BACKGROUND AND RELATED WORKS

This section introduces Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs). GANs and CNNs have been studied and used extensively; they have been used successfully in a range of diverse application areas including RF signal analysis. After introducing GAN and CNN we conclude this section with a discussion of existing emitter spoof-detection techniques that are relevant to this paper.

A. GENERATIVE ADVERSARIAL NET

GAN [4] is an unsupervised learning technique based on a synergistic application of ideas from game theory and machine learning. A GAN consists of two competing neural networks: a generator (which acts as an adversary or spoofer) and a discriminator (which acts as an adversary or spoofer detector), which are involved in an infinite horizon game. The input from a trained generator can be used to build smart and robust discriminative models that can operate in the presence of ''yet to be seen'', ''real'' adversaries. The overall training mechanism can be conceptualized as a ***min-imax game*** with two players, namely the generator and the discriminator. Each player improves through a sequential iterative training process. Theoretically the generator and discriminator are supposed to play the game indefinitely, but in reality. depending on the ratio of data and model density, the discriminator overpowers the generator in a finite amount of time, due to vanishing gradient of the generator. Note that the discriminator is generally deeper than the generator for a purposeful implementation of the GAN framework.

Following [4], to fit the generator's distribution p_g to the data x, a prior on input noise variables $p_z(z)$ to the generator is defined and a mapping to the data space is learned as $G(z; \theta_g)$, where G is a multi-layer perceptron with parameters θ_g. For the discriminator, the multi-layer perceptron is defined as $D(x; \theta_d)$, where $D(x)$ represents the probability that x was sampled from the data distribution, and θ_d is a scalar label. The objective of GAN training is to train D to maximize the probability of assigning the correct label (i.e., the detection problem) and train G to minimize $log(1 - D(G(z)))$, the probability of D making a correct decision on the fake input, simultaneously. Since G's cost depends on D's parameters and vice-versa, each player cannot control the other's parameters. This situation is best represented as a *min-max game* [8], with value function $V(G, D)$ formulated as follows: [4]

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{\sim p_x(z)}[\log(1 - D(G(z)))]$$

The objective of the GAN min-max game is to find the Nash equilibrium that minimizes the maximum loss. It must be pointed out that the GAN loss as defined above is hard to optimize and implementations typically use different strategies like pre-training of the discriminator to circumvent this problem. Readers interested in knowing more about GANs are referred to [4] for relevant details.

B. CONVOLUTIONAL NEURAL NETWORKS

Over the last decade neural networks (NNs) have become the tool of choice for applications of artificial intelligence (AI) in a variety of domains. They have been successfully used to build systems that have beat humans in tasks like image classification and language translation [4]. Though initially slow to adopt this new paradigm for implementing AI systems, the RF community has progressively embraced the use of NNs in the last couple of years with applications becoming mainstream by the day. Neural networks (NNs) are rapidly gaining ground in radio frequency machine learning (RFML) [2], [6] as they provide a way to build end-to-end systems that require minimal or no expert feature engineering. Inspired by their recent success, in this work we adopt neural networks for building a RFML system for the task of fingerprinting GPS constellations, using raw I/Q signal data with the goal of building a GPS agnostic positioning system. More precisely we show that convolutional neural networks (CNN) can learn GPS constellation signatures from raw I/Q data by exploiting the spatial correlations present in the same. Finally we show that these signatures can be used to identify specific locations on the surface of Earth.

CNNs are implemented using three types of layers (where each layer consists of several neurons): convolutional, pooling and fully connect layer. CNNs learn ''local features'' from the input (for example spatial correlation) through the use of specially designed structures called convolutional ''filters,'' which can be looked upon as masks that are designed to elicit special responses from the input data. To reduce the number

of features and elicit finer responses through aggregation, deep structures pool information from the *convolutional layers* for learning progressively finer features. These pooled features exploit the ''locality'' information in the input and in turn help the network to learn more compact and meaningful representations for solving a specific problem. The fully connected layers organize the nonlinear combinations of the high-level features from the CNN. CNNs have proven to be effective not only for image and video processing [5], but also for problems in the RFML domain. Lately, there has been quite a few attempts at using CNNs for learning different intrinsic RF parameters [6], [9], [10].

C. RELATED WORKS

This section discusses the design and implementation of effective spoofers for positioning systems using SDRs. We also discuss the state-of-the-art defense mechanisms against such spoofing attacks for various outdoor and indoor positioning systems. The advantages of using automatic feature learning techniques to prevent/circumvent such attacks over large-scale deployments is discussed. Finally, the section concludes with a discussion of artificial intelligence (AI) inspired spoof-detection techniques for modern location estimation services.

In 2008 Humphreys *et al.* [11] started to work on a portable civilian GPS spoofer to assess the impact of the threat of spoofing attacks on civilian GPS receivers. They also proposed two SDR-based and one cryptography-based measures for countering such attacks. However, all the methods studied and implemented were low impact and not without problems. Later a receiver-autonomous angle-of-arrival spoofing countermeasure was proposed in [12] by using a dual antenna receiver. It was argued that the presence of multiple antennas mounted at the receiver makes it difficult for the spoofer to successfully inject an artificial GPS signal into the system. An automatic gain control (AGC) based RF front end was proposed in [13] to enable spoofing detection in case a receiver was perceived to have been spoofed ''with varying degrees of success and computational complexity.'' However per their results, it was concluded that spoofing detectors should not rely solely on AGC though it can be used in conjunction with other techniques to provide a robust defense mechanism against spoofing attacks.

A multi-step defense against GPS spoofing was presented in [7], where the authors found that combining cryptography, signal-distortion detection and direction-of-arrival sensing, can provide a substantially secure countermeasure that could be commercially deployed. In [14], the authors implemented a practical real-time GPS spoofing attack for the use case of road navigation. They argued that there were no effective countermeasures for protecting civilian GPS systems against spoofing attacks until that time. Tanil *et al.* proposed a novel Kalman filter based inertial navigation system (INS) monitor in [15] to detect GNSS spoofing attacks on an aircraft, even when the spoofer was able to correctly estimate the real-time position of the aircraft. They demonstrated the applicability of their proposed approach by stopping a spoofing attack on a Boeing 747. In [16], the authors presented a RF fingerprinting based method to detect pseudo-GPS signals by leveraging the Doppler frequency shift in the signal as a ''fingerprint''.

Even with the large scale deployment of GPS and other worldwide positioning services through satellites, indoor positioning suffers due to the low reachability of satellite signals. GPS signals are hardly discernible in indoor settings due to the use of low power transmissions using spread spectrum. However, due to the advent of ubiquitous computing, recent efforts seek to make indoor positioning fool-proof and accurate. In [17], Randall *et al.* presented an indoor positioning system using carpet-like distributed Radio Frequency Identification (RFID) tags to estimate location. A remote navigation system by estimating the indoor floorplan was proposed in [18] for RFID-augmented environment. On the other hand, a hybrid location approach, which opportunistically selects between signal strength–based fingerprinting techniques for indoor positioning and traditional GPS-based positioning for the outdoor localization, was proposed in [19], for mobile devices which frequently move between indoor and outdoor settings.

In [20] the authors explored the idea of using multiple RF sources for indoor positioning. They showed that it is possible to compute accurate indoor positioning information if one can choose between different RF sources when computing the location information using nearest neighbor based methods on the space of RF fingerprints. Their method protects against spoofing attacks since in order for the spoofer to work all the RF sources need to be spoofed, a task that is hard to achieve if the number of sources is large. Furthermore the method can be deployed with acceptable real time performance.

Another real-time indoor localization technique based on dynamic measurement compressive sensing for wireless sensor networks was proposed in [21]. Similarly, in [22] the authors proposed an indoor location and orientation estimation technique by using the user's 3D location and heading direction via a simple trilateration algorithm. However, though all these indoor localization techniques have resulted in better indoor position estimation techniques, they have not dealt with the fact that malicious entities can still play havoc with such systems in the absence of active spoofing mitigation schemes. Moreover with the large scale deployment of indoor positioning systems and the wide availability of ubiquitous computing, it has become important to deploy end-to-end spoofer detection systems in indoor environments. Thus there is a need for research into building AI-based spoofer detection and mitigation techniques that can scale to large deployment areas in both indoor and outdoor settings.

III. PROPOSED GANSAT FRAMEWORK: SPOOF-DETECTION AND LOCATION PREDICTION

This section presents details of the proposed GANSAT framework (shown in Fig. 3), which consists of a system for detecting spoofed GPS signals and subsequently using the filtered ''real GPS'' signals to estimate the approximate location of

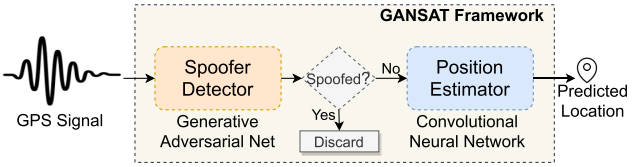


FIGURE 3. The system architecture of GANSAT framework.

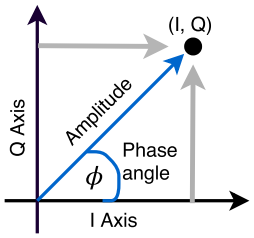


FIGURE 4. Amplitude and phase of IQ data.

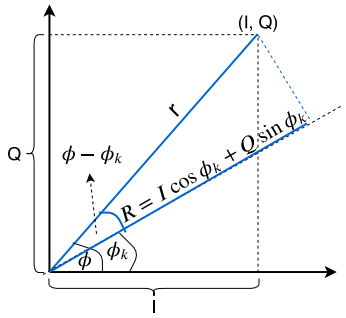


FIGURE 5. Computing the multi-angular projection R for phase ϕ_k.

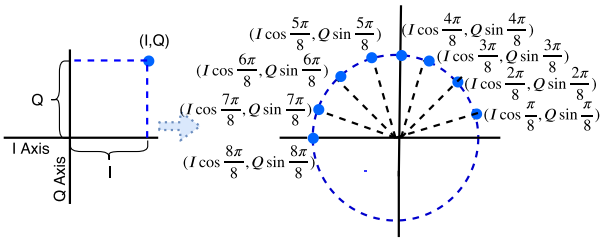


FIGURE 6. Phase angles for computing multi-angular projection.

the receiver using the idea of satellite constellation signatures. Note that as mentioned before, the GANSAT implementation passively senses GPS signals without the need for the receiver to get a "lock" and hence works irrespective of the signal strength at a given location. To describe the framework, this section first introduces the idea of a "multi-angular projection." This is used with the raw GPS signal data to enhance the spatial correlation present in the data by projecting it to a higher dimensional space. In multi-angular projection, the amplitude of the in-phase (I), and the quadrature-phase (Q) components of the raw signal data are projected to a higher dimensional space. The GAN-based approach (first part of GANSAT framework) filters the `spoofed` signals from `real` ones using this high dimensional representation of the raw GPS data. Finally, the high dimensional representation of the filtered `real` signals are fed into a CNN model (second part of GANSAT framework) for location estimation, using implicit "constellation fingerprints" learned from the training data.

A. MULTI-ANGULAR PROJECTION

Raw signal data including that from GPS satellites consists of multiple I and Q value pairs. Note that a single (I, Q) pair is a point in the 2D complex plane and is associated with an amplitude r and a phase ϕ (see Fig. 4). Fig. 4 illustrates that $I = r \cdot \cos(\phi)$ and $Q = r \cdot \sin(\phi)$ and hence the complex number (I, Q) can be represented by $(\cos(\phi), \sin(\phi))$ when the amplitude r is a constant. Note that a complex number is fully determined by a radial coordinate, r, and an angular coordinate, ϕ. This in turn means that if one of these parameters is known, the complex number is fully determined by the other remaining parameter. Thus if r is fixed, then the complex number is determined by the phase ϕ. This angle denotes a direction with respect to the horizontal coordinate axis and hence the complex number in this case is synonymous with a specific direction. Thus each point in the complex plane determines a direction specified by phase of the complex number corresponding to the point.

Given a set of (I, Q) values for a time varying signal, denoted by $(I_j, Q_j), j = 1, 2, \ldots, N$, a multi-angular projection is obtained by projecting each (I_j, Q_j), on a set of pre-determined directions specified by a set of complex numbers having phase angles ϕ_k, where $k = 1, 2, \; ldots, M$. The projection of (I_j, Q_j) in the direction specified by ϕ_k defines another complex number $P_{(I_j,Q_j)}$ having phase ϕ_k and amplitude $I_j \cdot \cos\phi_k + Q_j \cdot \sin\phi_k$. Note that the set of phase angles for computing the projections for each $j = 1, 2, \; ldots, N$ are the same. Hence for each raw I/Q data (I_j, Q_j), we consider the set of amplitudes of the resulting projections along the M directions, as the input feature set. This is a vector in the M-dimensional real space. Thus for each (I_j, Q_j) $j = 1, 2, \ldots, N$, the resulting input feature is $(I_j \cdot \cos\phi_1 + Q_j \cdot \sin\phi_1, \ldots, I_j \cdot \cos\phi_M + Q_j \cdot \sin\phi_M) \in \mathcal{R}^M$. It should be clear that the multi-angular projection can be interpreted as a dimension expansion technique which captures the variations present in the input data by projecting the data along different directions specified by the phase angles resulting in a high dimensional projection. Fig. 5 shows the details of the multi-angular projection method for a single (I, Q) value and direction ϕ_k. For our implementation, we used eight phase angles for computing the multi-angular projection and these angles are shown in Fig. 6.

It must be noted that unlike dimension reduction techniques that are used in machine learning, the multi-angular projection is a dimension expansion technique. While it may seem counter-intuitive, the use of dimension expansion techniques is not without precedent in the literature. In fact, one of

the classical strategies in machine learning is to project points to a higher-dimensional feature space in order to learn the decision boundary in cases where the points are not separable in a lower-dimensional space. For example, 2D points, which can be separated by a circular decision boundary and hence are not linearly separable in two dimensions, can be projected onto the surface of a 3D parabola where they become linearly separable [23]. The intuition behind this idea (which is generally called ''kernelization'') is that projecting the 2D points on to the surface of the 3D parabola provides a new ''perspective'' for the data, which can be exploited to learn a linear decision boundary. The generalization of this approach can be found in the idea of the ''kernel trick'' where the classifier is learned in a higher dimensional space in order to exploit hidden properties of the input data.

The inspiration for the multi-angular projection comes from this idea of kernelization and the fact that the I/Q values obtained from the GPS satellites (or as a matter of fact any transmitter) have inherent properties that can be exploited to fingerprint the transmitters [2]. These fingerprints can be used to distinguish between real and fake signals transmitted by (or purported to be transmitted by) a given emitter. Furthermore, since the properties of the emitted signals are also affected by the intrinsic environmental noise, a combination of fingerprints from multiple emitters can be correlated to the operational environment of the emitters (and hence used for location inference). The idea of characterizing the location using signatures of signals emitted from multiple transmitters is the idea of ''emitter constellation fingerprinting.'' However multi-path effects and random variations in the environmental noise make it hard to learn these characteristics. This is especially true in the case of GPS signals, due to its use of spread spectrum for transmissions [24]. Under these circumstances, it becomes non-trivial to directly decipher and learn these inherent characteristics of the signals for the purpose of filtering out real GPS signals from fake ones or for the task of location inference using emitter constellation fingerprinting. Thus in order to decouple the effects of multi-path and random variations of environmental noise from the received GPS signal using multi-angular projection, the received signal data is projected to a higher dimensional space. This is done to implicitly learn the decision boundaries by decoupling these distortions present in the data from each other. Intuitively, the projection should provide enough ''perspectives'' of the data for the learning algorithm to consider and in the process, the algorithm can be expected to learn pertinent feature representations for the given task. The effectiveness of the proposed projection technique is analyzed through experimentation during the implementation of the location inference system in Section V-C and V-D.

B. GAN FOR SPOOF-DETECTION

As mentioned before, a GAN framework consists of two distinct models: the generator ($\mathcal{G}$) and the discriminator ($\mathcal{D}$). The goal of the generator is to create ''fake'' samples that are akin to the real ones by implicitly learning the real data distribution. The goal of the discriminator is to differentiate the fake data from the ''real'' ones by estimating the probability that a given sample was obtained by sampling the real data distribution rather than using $\mathcal{G}$. As GANSAT is based on the GAN framework, it consists of both these entities, trained with the express purpose of generating fake GPS data and identifying the same. In GANSAT, during the training phase, the generator acts as a malicious GPS emitter, which generates fake GPS data with the goal of forcing the discriminator into making a mistake (accepting the fake data as real). The discriminator in turn aims to minimize the probability of error in classifying real GPS data from fake ones. During the training phase, the decision of the discriminator is conveyed to the generator, which uses this information to fine tune its parameters to generate more realistic ''fake'' data. At the end of the training, the generator learns to generate ''fake'' or spoofed GPS signals, and the discriminator learns to distinguish between the real and spoofed GPS data.

1) GENERATIVE MODEL

The generative model works by accepting a random vector as input and generating ''fake'' GPS data as output by sampling from an approximation of the real data distribution that it learns implicitly through unsupervised learning. Note that in the context of the GANSAT application scenario, which is that of generating I/Q data spoofing raw GPS signals, the generator can be thought of as generating an N class output, where N is the number of I/Q values in the sample corresponding to a time t. Thus the generator input is a $1 \times N$ random vector where each component is two dimensional. This is conditioned on a prior distribution and the output is a vector of I/Q samples of the same dimension but sampled from the latent distribution of the real GPS data, which is denoted by $p_g(g)$. In effect the generator creates a signal using a random input which can be modeled as consisting of two additive components the first of which is $s(t)$, which is defined as:

$$s(t) = c_1 m(t) + c_2 r(t) + c_3 l(t) \tag{1}$$

Here $s(t)$ is modeled as a continuous time varying signal modulated onto a sinusoid with varying frequency, phase, amplitude or some permutation of these parameters. In the above expression, $m(t)$, $r(t)$, and $l(t)$ are the time series continuous signals for modulation, amplitude, and phase respectively. These are usually selected randomly using some prior imposed on the data that factors in the knowledge about the underlying communication system. The coefficients c_1, c_2, and c_3 are some path loss or constant gain terms associated with $m(t)$, $r(t)$, and $l(t)$ respectively. The input to the generator, $g(t)$, is finally obtained as:

$$g(t) = s(t) + n(t) \tag{2}$$

where $n(t)$ is Gaussian white noise. $g(t)$ is used as input to the generator which generates a ''fake'' data point from this input by sampling from the latent probability distribution $p_g(\mathbf{x})$ over sample space ($\mathbf{x}$) of the real I/Q data. The distribution

$p_g(\mathbf{x})$ over sample space $\mathbf{x}$ is learned implicitly during training. Note that we have indexed the prior information, noise and the generated signal by time t in order to acknowledge that the signal characteristics can change over time. However, for this work, we do not explicitly consider the effects of the variation of signal characteristics and noise with time. The goal of the GAN is to minimize the cost $V(\mathcal{G})$ which is the probability that $\mathcal{D}$ correctly identifies the signal as being generated from $\mathcal{G}$.

2) DISCRIMINATIVE MODEL & GAN TRAINING

The discriminator is trained with both *real* signal data (x) drawn from the data distribution $p_{data}(x)$ and *generated* signal data from the generator ($g(t)$). The discriminator's objective is to successfully identify real data samples from fake ones by maximizing the probability of classifying real samples as such and minimizing the probability of classifying fake samples as real. Thus the objective of the overall GAN model is formulated as follows: the goal of the generator is to maximize the probability that a fake sample is classified as real while that of the discriminator is to maximize the probability of classifying real samples as such and minimize the probability of classifying fake samples as real. Let the overall GAN cost be denoted by $V(\mathcal{G}, \mathcal{D})$ and let $\mathcal{D}(x)$ be the probability of classifying a sample x as real. Then the GAN cost function can be written as $V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{p_{data}(x)} \log \mathcal{D}(x) + \mathbb{E}_{p_g(g)} \log(1 - \mathcal{D}(\mathcal{G}(g)))$, where $p_g(g)$ is the generator's distribution over the *generated* signal samples g and $p_{data}(x)$ is the distribution of *real* signal samples x. Note that $\mathcal{D}(x)$ is the probability that x came from $p_{data}(x)$ rather than from $p_g(g)$ and $\mathcal{D}(\mathcal{G}(g))$ represents the probability a fake sample g generated from $\mathcal{G}$ is classified as real, that is coming from $p_{data}(x)$. Finally the goal of the GAN training is:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) \tag{3}$$

After training, the GAN framework eventually converges to an unique optimal discriminator for a given generator, $\mathcal{D}^*(\mathbf{x}) = \dfrac{p_{data}(\mathbf{x})}{p_{data}(x) + p_g(g)}$. It is clear that $\mathcal{D}$ is optimal when the discriminator can distinguish between every *real* signal data (x) and *generated* signal data (g). Similarly, it is also clear that $\mathcal{G}$ is optimal when $\mathcal{D}$ cannot distinguish between x and g, that is, $\mathcal{G}$ is optimal when $p_g(g) = p_{data}(x)$.

3) PROPOSED GAN ARCHITECTURE

In the proposed GAN architecture, the generator first generates data from the prior constrained random noise without any information about the latent data distribution; but after a few training epochs, $\mathcal{G}(g)$ learns to generate signals from the latent data distribution $p_g(g)$ using feedback on the performance of the discriminator which implicitly provides information on the latent data distribution. Note that an epoch consists of a forward and a backward pass of the model over the entire dataset. On the other hand, the trained discriminator learns to differentiate between the *generated* data distribution ($p_g(g)$) and the *real* data distribution ($p_{data}(x)$) over the different training epochs. In effect the trained model tries to distinguish the *generated* signals (g) from the *real* ones (x). The *Adam* [25] optimizer is used for the GAN training with different *learning rates* per epoch. The output from the discriminator is fed back to both $\mathcal{G}(g)$ and $\mathcal{D}(g)$ as a form of implicit information about the latent data distribution (for the generator) and generator's distribution (for the discriminator), which in turn helps them to adjust their respective models to better compete against each other.

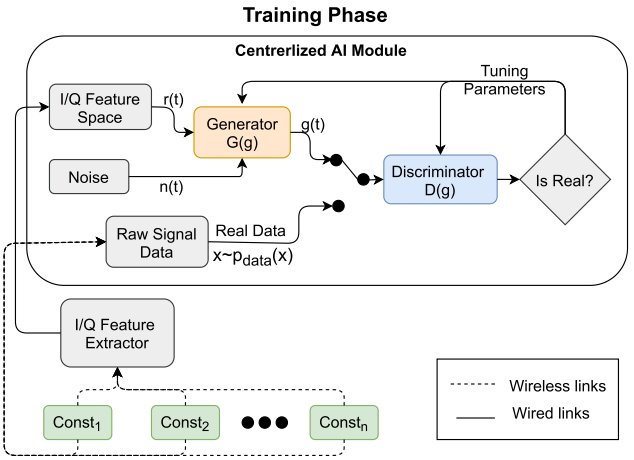


FIGURE 7. GAN training in the GANSAT framework.

4) GAN IMPLEMENTATION

Our GANSAT implementation is compatible with data generated by any existing GPS system. The GAN trained discriminator is deployed inside the GANSAT system for the purpose of spoof-detection and it is responsible for filtering out real GPS data from spoofed ones. Note that the discriminator model though trained with the generator input is used to filter out GPS data spoofed by an actual hardware spoofer and not data generated by the generator, which is used only during GAN training. During training, the discriminator is trained with data from the *generator* and *real* GPS satellites. The training methodology is presented in Fig. 7 and occurs over several rounds (epochs) with the generator and discriminator both contending to achieve superiority over the other. As mentioned before, during training, the output from the discriminator is used as input to both the generator and the discriminator, and this feedback is used to tune the hyper-parameters of the system. Over time, the discriminator eventually overpowers the generator. Once the GAN training is complete, the trained discriminator is deployed in the AI module of the GANSAT framework, conceptually presented in Fig 8.

The details of the GAN implementation is shown in Fig. 9. The generator consists of three *dense* layers with neurons in the scale of the *sample size*. The sample size is implementation specific, and for these RF experiments, we used a *sample size* of 1024. In the beginning, the generator starts by computing on random noise, but with the progress of GAN

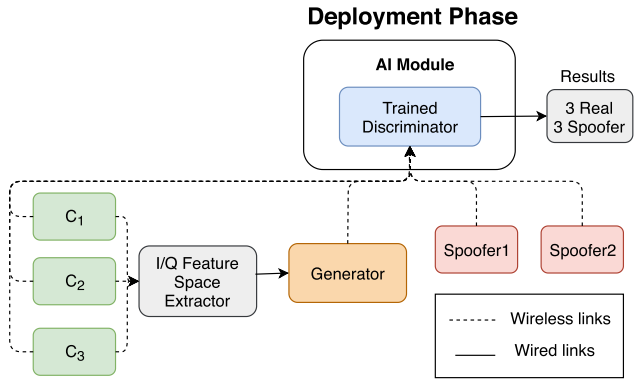


FIGURE 8. Deployment of trained discriminator in spoofed GPS environment.

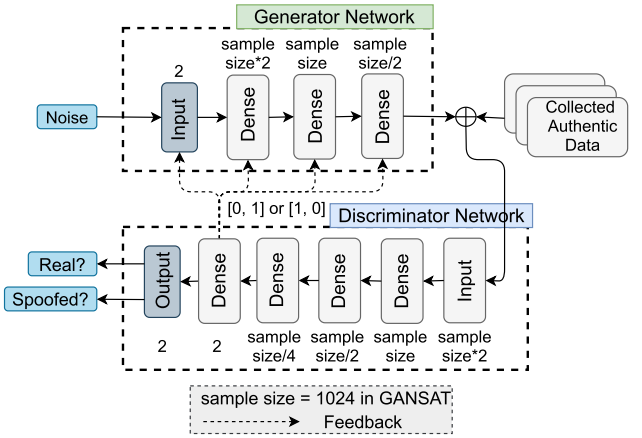


FIGURE 9. The details of GAN implementation in GANSAT.

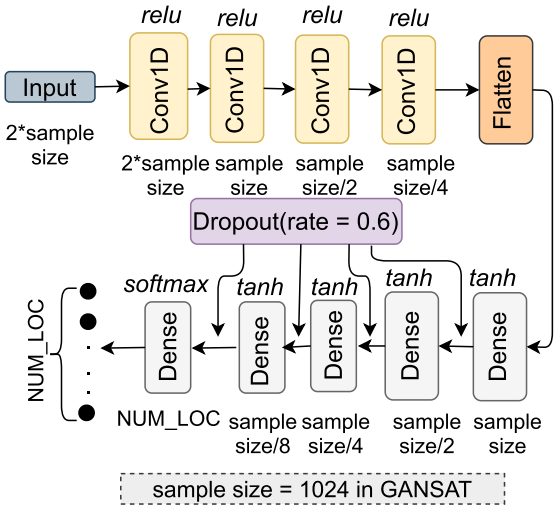


FIGURE 10. The proposed CNN architecture of GANSAT for GPS satellite constellation fingerprinting.

TABLE 1. Parameter setup for each convolution layer.

Layer	#Filters	Kernel Size	Strides
1st Conv Layer	2048	1	1
2nd Conv Layer	1024	4	2
3rd Conv Layer	512	2	2
4th Conv Layer	256	2	1

training, it keeps getting feedback from the discriminator and this feedback acts as a form of implicit information on the latent distribution of the real GPS data. This helps the generator adjust its parameters and generate better quality outputs which is more akin to the real GPS data. The *generated* samples are of length $2 \times$ *sample size* and are sampled from the estimated latent distribution $p_g(g)$. The discriminator $\mathcal{D}$ consists of one input layer having $2 \times$ *sample size* neurons, three hidden layers with neurons in scale of the *sample size* and finally a *softmax* output layer with 2 neurons to classify an input as either `spoofed` or `real`. GANSAT uses tanh activation at the hidden layers along with a *Dropout* [26] of 0.5 in between these layers for regularization. Note that during the training phase, the discriminator is trained on *real* and *generated* signals, whereas in the deployment phase the discriminator is able to distinguish between *real* and *hardware spoofed* (as well as *generated*) signals.

C. 1D-CNN FOR LOCATION PREDICTION

In order to learn the intrinsic properties of GPS signals, GANSAT exploits the spatial correlation present in the raw GPS signal data using a CNN. Since the GPS signal is received as a stream of I/Q values, locality is inherently present in the data and thus intuitively the spatial correlation in the data can be best exploited through the use of *1-D convolution* in the CNN models [5]. The GANSAT design and implementation uses a CNN with four *conv1D* [27] layers, a *Flatten* operation and five fully connected (*dense*) layers, as shown in Fig. 10. The number of filters used for each layer (shown in Fig. 10) is in the scale of *sample size*. A *Dropout* [26] of 0.6 after each *dense* layer is used for regularization and Table 1 presents information about kernel and stride sizes for each convolution layer in our testbed implementation. GANSAT uses the *ReLU* [28] activation for all the convolution layers and tanh activation [5] for all fully connected layers, other than the last layer which uses the *softmax* [5] activation. Detailed settings of the configuration parameters are presented in Table 2. The system employs *stochastic gradient descent* (with learning rate of 10^{-4}) based optimization with categorical cross-entropy training. During training, we set the maximum epoch to 100 with an early stopping condition such that if there is no improvement of validation loss is observed for five consecutive epochs, then the training is stopped. We observed through multiple training runs that the proposed model converges within a range of 45-60 epochs.

It is to be noted that we designed the CNN with only 4 convolution layers and 5 fully connected layers for faster training [29] and also because no significant increase in the testing accuracy was observed after increasing the number of layers.

1) OPTIMIZATION IN THE PROPOSED 1D-CNN

As discussed in Section III-A, the input to the GANSAT system is the output of the multi-angular projection applied

on the raw GPS signal data as a pre-processing step. The result of the projection operation is represented as $\eta = (I_j \cdot \cos\phi_1 + Q_j \cdot \sin\phi_1, \ldots, I_j \cdot \cos\phi_M + Q_j \cdot \sin\phi_M) \in \mathcal{R}^M$, which is the input to the proposed NNs (both the CNN as well as the discriminator). For an input η, the output of the k^{th} convolution layer of is:

$$\mathcal{C}_k = \mathcal{P}(ReLU(W_k^c * \mathcal{C}_{k-1} + b_k^c)), \quad k = 1, 2, 3, 4, 5 \quad (4)$$

where $\mathcal{C}_0 = \eta$, "$*$" is the convolution operation, W_k^c and b_k^c are the convolution kernel and bias corresponding to the k^{th} layer and $ReLU(\cdot)$ denotes the activation function. $P(\cdot)$ is the zero-padding operator to fill the borders with zeros for effectively using the convolution operation. Next the output of $\mathcal{C}_5$ is transformed to a flattened tensor η_f to be fed into the fully connected layers (or *dense* layers). Here the output of l^{th} hidden dense layer is given by:

$$\mathcal{D}_l = \tanh(W_l^d \circ \mathcal{D}_{l-1} + b_l^d), \quad l = 1, 2, 3, 4 \quad (5)$$

where $\mathcal{D}_0 = \eta_f$, '$\circ$' is the matrix multiplication operation while W_l^d and b_l^d are the weight and bias corresponding to the l^{th} dense layer respectively and $\tanh(\cdot)$ denotes the activation function. Finally the output layer computes a feature vector z from the input η as follows:

$$z = [z_0 z_1 z_3 \cdots z_v \cdots z_{C-1}] \quad (6)$$

where C is the total number of classes (here locations) and v is the vector index. The *softmax* function on z predicts the label of the input η. If the original label of input η is ζ, then the predicted label is given by:

$$\hat{\zeta} = arg\max_v \frac{e^{z_v}}{\sum_m e^{z_m}}, m = 0, 1, 2 \cdots (C-1); \quad v = 0, 1, 2 \cdots (C-1) \quad (7)$$

The goal for training the proposed 1D-CNN is to find the parameters α that minimizes the cost function $J(\alpha)$:

$$arg\min_\alpha J(\alpha) = arg\min_\alpha \sum_{i=1}^{N} J_i(\alpha) = arg\min_\alpha \sum_{i=1}^{N} (\hat{\zeta} - \zeta)^2 \quad (8)$$

where N is the total number of input samples.

D. 2D-CNN FOR IMPROVED LOCATION PREDICTION

The location prediction system can be improved with a two dimensional (2D) convolution instead of a 1D-convolution as described above. In the 2D-CNN approach each projected sample is treated as an image of dimension (8, *sample size*). This simple change made the location prediction system more accurate and simpler while decreasing the training time. The details of the proposed architecture with 2D convolution is presented in Fig. 11. GANSAT uses 64 filters in the Conv2D [27] layer with (2, 2) kernel size and (1, 1) stride with ReLU [28] activation. The max pooling layer has a pool size of (2, 2) and stride (2, 2), while other details are provided in Fig. 11 and Table 2. The *dense* layer, after *Dropout* [26] and *Flatten* operation, uses ReLU [28] activation as well. It is evident that the 2D-CNN architecture is shallower compared to the 1D-CNN implementation. Another advantage of 2D-CNN approach is that one does not need to change the number of filters depending on the variable *sample size*, as is required for the 1D CNN.

TABLE 2. Details of parameter setup for CNN Model of proposed GANSAT framework.

Parameters	Values
Input Layer	2× sample size
Output Layer	Number of Locations
Weight Initializer	Xavier Normal
Weight Regularizer	L2
Activation Function for Convolution Layers	ReLU [28]
Activation Function for Dense Layers	tanh
Output Activation Function	Softmax
Dropout Rate for GAN	0.5
Dropout Rate for 1D-CNN	0.6
Dropout Rate for 2D-CNN	0.2
Cost Function	Cross Entropy
Optimizer	*stochastic gradient*
Nesterov Momentum	0.9
Decay Factor	10^{-6}
Learning Rate	10^{-4}
Epochs	100 (w early stopping)
Batch Size	128

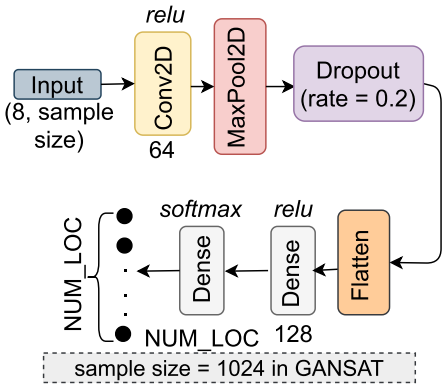


FIGURE 11. The proposed 2D CNN architecture of GANSAT for improved location prediction.

1) OPTIMIZATION BY THE PROPOSED 2D-CNN

The overall optimization technique for the 2D CNN is similar to the 1D-CNN. The primary difference is the lesser number of convolutions and fully connected layers. For an input η, the output of the 2D-convolution layer is given by:

$$\mathcal{C}_1 = \mathcal{P}(ReLU(W_1^c * \eta + b_1^c)) \quad (9)$$

where $*$, W_1^c, b_1^c, $ReLU(\cdot)$, and $P(\cdot)$ represent the same operations as presented earlier. The computation at the single hidden dense layer $\mathcal{D}_1$ is given by:

$$\mathcal{D}_1 = ReLU(W_1^d \circ \eta_f + b_1^d) \quad (10)$$

where η_f, $\circ$, W_1^d, and b_1^d represent the same things as before. However there is no change in the *softmax* activation function for the output layer. Finally, the goal of optimization for the

proposed 2D-CNN can be summarized as:

$$arg\min_{\alpha} J(\alpha) = arg\min_{\alpha} \sum_{i=1}^{N} J_i(\alpha) = arg\min_{\alpha} \sum_{i=1}^{N} (\hat{\zeta} - \zeta)^2 \tag{11}$$

where as before N is the total number of training examples.

IV. TESTBED SETUP

In order to validate the proposed GANSAT framework, we collected raw GPS signals in real outdoor environments at five different locations in the Florida panhandle area of USA, at different times of day: (a) Shalimar a small town in the Okaloosa county Florida; (b) Crestview, another small town about 25 miles away from Shalimar; (c) Defuniak Springs, a small locality along the Interstate 10 corridor of Florida; (d) Freeport, a small town off Highway 20 in the Florida panhandle; and (e) Niceville, a mid sized town along around the Eglin Air Force Base, Florida. An outdoor location was selected randomly in the aforementioned towns and our equipment was setup at that location for data collection. Before collecting the data, we noted the latitude and longitude of the selected location for records. We would like to mention that we selected these five locations and no more due to the difficulty in collecting the data by driving to different locations while at the same time ensuring the safely and security of the people involved. The data was collected during the Covid pandemic that is still raging across the world and we had to ensure proper precautions as specified by the Universities. Though we wanted to visit other locations and collect more data, we finally decided against it due to health and wellness concerns of the researchers. However, since the locations were randomly selected, the results generalize to any location on the planet from where the GPS constellation is visible. Note that at some places (and times), the GPS signal was strong and we obtained a lock and NAV messages, whereas at others, the signal was weak and we did not get a lock or NAV messages. We also collected fake GPS data inside an uncontrolled laboratory environment, spoofing the exact same 5 locations (latitude and longitude wise). We implemented the generator and discriminator (of the GAN) to detect and filter out the `spoofed` signals from `real` ones and then used a CNN to predict the locations from `real` GPS signals in *GPS-deprived* or *weak-GPS* environments. The details of the experiments are discussed below.

A. REAL GPS DATA COLLECTION

In order to collect the raw over-the-air GPS signals, we used an universal software radio peripheral (USRP) software defined radio (SDR), namely B210 from Ettus Research [30]. To collect data using the B210 SDR we used the open source "Global Navigation Satellite Systems" software-defined receiver (GNSS-SDR) [31] package which was installed on a 2.9 GHz 16 GB Intel i7 system. Our data acquisition system utilized a B210 compatible GPS Disciplined Oscillator (GPSDO-TCXO) and an active GPS antenna on the USRP-B210 SDR. Since we used an active GPS antenna, we placed a bias-T between the USRP and the antenna to connect the antenna to a power source, so that the required voltage (5V) could be delivered [32] to the antenna without any overload.

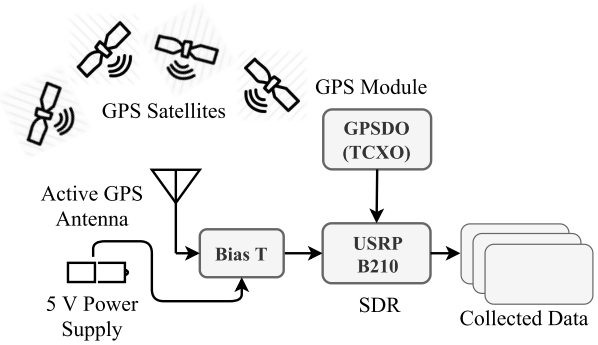


FIGURE 12. An overview of GPS data collection.

TABLE 3. GPS receiver configuration parameters.

Parameters	Values
Data Type	Complex
Sampling Frequency	4 MHz
Channel Frequency	1575.42 MHz
Antenna Gain	50 dBi
Antenna Port	RF-A: RX2
Antenna Height	8.5 ft
(I, Q) pairs	$>$ 50 Million

The data collection setup is shown in Fig. 12 and the configuration parameters are shown in Table 3. We tuned the radio to 1575.42 MHz as we wanted to capture the Coarse-acquisition (C/A) and the L1 civilian (L1C) codes for the L1 GPS band. Note that during our data collection in strong-GPS environments, GNSS-SDR was able to get a *Position Fix*; however for *GPS-free* and *weak-GPS* environments we observed a consistent loss of satellite locks and hence the receiver was not able to get a position fix in these cases. For these cases we simply collected the raw GPS I/Q data and inferred the location from a smartphone based GPS.

In order to collect GPS signals using GNSS-SDR, a small script was developed with configuration parameters that specify the B210 signal source and the Civilian L1 C/A Band decoding process. During the execution of the script, GNSS-SDR attempts to lock onto and track the GPS transmitters in the visible satellite constellation at a given location. Finally the GNSS timer starts to get bit synchronization with each of the visible satellites and obtain *bit sync* messages to receive accurate navigation information. After 2-3 minutes, gradually it starts to receive the *NAV* messages from the visible satellites. At this point it is able to compute the receiver's coordinates by getting a *Position Fix* after receiving a stream of *NAV* messages.

The GPS data collection setup is shown in Fig. 13. Note that we collected raw over-the-air GPS signal data in five different outdoor locations as shown in Fig. 14. All the locations

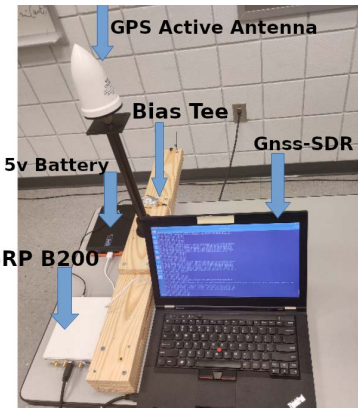


FIGURE 13. The GPS data collection setup in our lab.

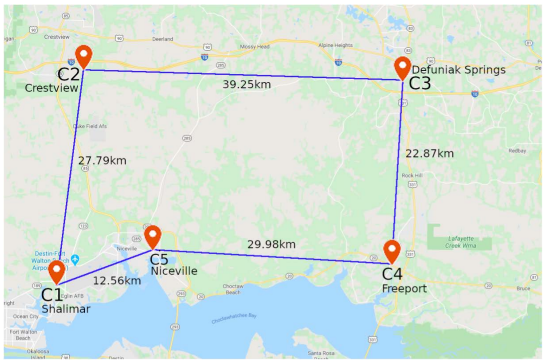


FIGURE 14. The five locations of data collection in northern part of florida.

TABLE 4. Information about locations and time of real GPS data collection.

Locations	Coordinates	Date-Time	Type
Shalimar, FL	(30.4749239,-86.5729871)	01/28/2020, 17:17:00	*weak-GPS*
Crestview, FL	(30.7158708, -86.5383062)	01/28/2020, 22:09:30	*GPS-deprived*
Defuniak Spring, FL	(30.7046011, -86.1232538)	02/03/2020, 00:55:42	*GPS-deprived*
Freeport, FL	(30.4981759, -86.1367235)	02/03/2020, 00:11:20	*GPS-deprived*
Niceville, FL	(30.5137804, -86.4484978)	02/03/2020, 18:06:38	*strong-GPS*

were a minimum of 10 miles apart from each other and we show the exact latitude, longitude and GPS signal strength at each of these locations in Table 4. We collected most of our data within two days, at different times in order to get unique visible GPS satellite constellations per location.

In Shalimar, we were able to lock onto the GPS satellites intermittently, obtaining the receiver's coordinate for a period of 1-2 seconds only; hence, we label this location as having a *weak-GPS*. In Niceville, we observed that the GNSS-SDR was able to get the lock and obtain the receiver's coordinates consistently; hence we label it as a *strong-GPS* location. However for the other three places we observed that GNSS-SDR was unable to obtain the lock consistently and hence we labeled these locations as *GPS-deprived*, since the receiver was unable to calculate it's coordinate even once. We chose a combination of all three types of environments to show that the efficacy of the proposed approach is the

(a) with Mounted GPS Antenna on Car

(b) for a Fixed Location

FIGURE 15. The real GPS data collection setup.

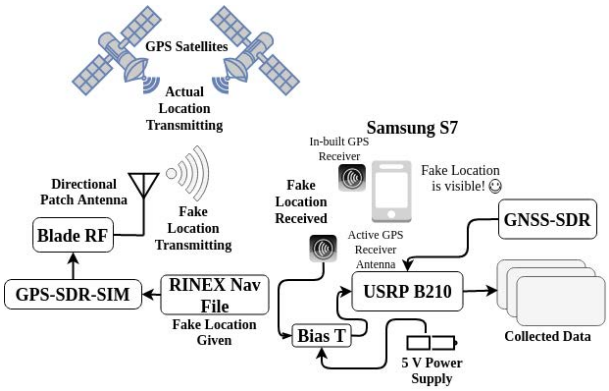


FIGURE 16. An overview of the designed spoofer.

same irrespective of the strength of the received GPS signal. We show the actual positions as obtained from Google Maps using Google high accuracy mode in Table 4. Our data collection setups are shown in Fig. 15.

B. GPS SPOOFING ATTACK IMPLEMENTATION

Software-defined GPS transceivers were a natural choice for studying civilian GPS spoofing and its effects. In a software defined GPS transceiver, the real-time correlators, tracking loops, and navigation solvers are all implemented in software on a programmable processor.

In order to spoof GPS signals, we use the open source GPS-SDR-SIM [33] package. The SIM package accepts a Receiver Independent Exchange Format [34] (RINEX) version 2 navigation file as input, with information about the latitude and longitude to be spoofed being provided by the user. It then produces fake GPS samples based on prior information about GPS satellite positions at the given latitude, longitude.

Once the fake GPS data file is generated by the software, it is transmitted via an SDR, namely the Blade RF 2.0 micro xA4 [35] manufactured by Nuand. The samples were transmitted on a frequency of 1575.42 MHz with a sampling rate of 2.6 Mhz and bandwidth of 2.5 Mhz. In order to test whether the samples could be used by a real GPS receiver for position computation, we collected these samples using the inbuilt GPS receiver of a Samsung Galaxy S7, operating in GPS only mode. An overview of the spoofing process is depicted in Fig. 16. The GPS data was then used by the "GPS Test" application [36] which was loaded onto the phone in order to parse the location information from the data. The results

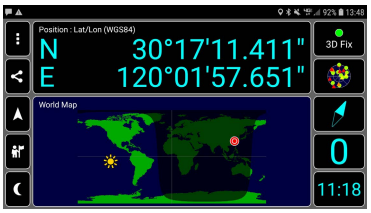


FIGURE 17. Screenshots of standard GPS vs. spoofed.

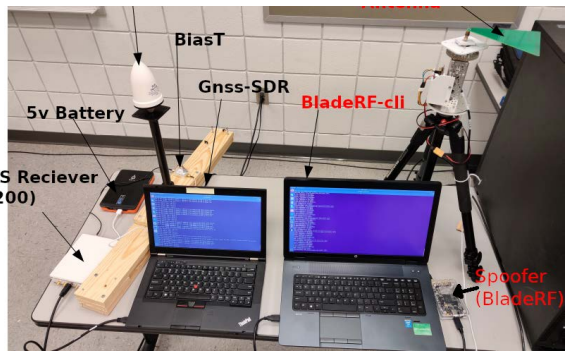


FIGURE 18. The spoofer setup in lab environment.

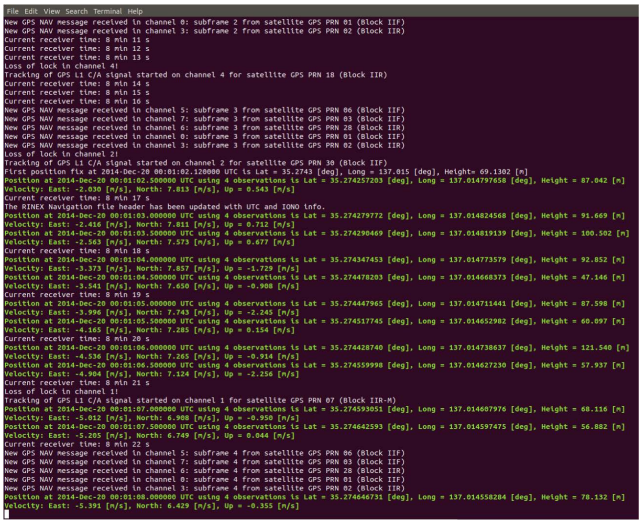


FIGURE 19. A screenshot of how GNSS is locking onto designed spoofer.

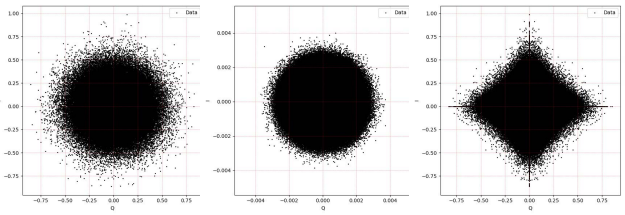
obtained are shown in Fig. 17. We performed multiple tests by varying the distance between the transmitter and receiver from 1 to 15 ft. For each of these tests we used the GPS Test [36] application to verify that the correct latitude and longitude were being spoofed and that the same was being received by the receiver. As can be seen from Fig 17, which shows the results of one of our spoofing experiments, even though the actual location of the receiver was in Florida, USA, the receiver was getting latitude and longitude information of some location in China which was being broadcast by the designed spoofer.

Once we confirmed the effectiveness of the spoofer, we collected spoofed raw GPS signal data using an USRP B210 running GNSS-SDR as mentioned in Section IV-A. The spoofed data collection setup is shown in Fig. 18. A sample output from GNSS-SDR while being spoofed with a file generated from ephemerides recorded in 2014 is shown in Fig. 19. It is clear from Fig. 19 that GNSS-SDR was successfully obtaining a ''lock'' with the spoofed data and calculating fake latitude and longitude (shown in green) after getting constant *NAV* messages. Thus, though the GPS receiver should have calculated the actual latitude and longitude of the position of the receiver (which is the latitude and longitude of our lab), due to the presence of the spoofer, it ended up computing the latitude and longitude of the spoofed location transmitted by the spoofer. This fake raw GPS signal data collected from the spoofer is used as input to the GAN trained discriminator model during the testing phase.

C. COLLECTED DATASETS

We collected both real and fake raw GPS data for 5 minutes at each of the five locations selected for our experiments. We obtained more than 50 million I/Q samples for each of the datasets (real and fake). Though we collected a variable number of training samples at each location, we used the same number of samples during GANSAT training for all the locations to mitigate the data skewness problem observed in ML. As mentioned in Section IV-A, the GNSS-SDR setup was tuned to the L1 GPS band and raw GPS I/Q samples with two features, I and Q, were obtained for each of the locations at any given time during the data collection process. Note that we did not *parse* the GPS data packets even if a lock was obtained by the receiver. We simply dumped the raw I/Q values received at the receiver at a given time at any given location. These raw I/Q samples are nothing but *complex64* numbers stored in binary files. The two types of collected datasets are:

- *Real GPS Dataset:* We collected more than 250 million I/Q samples from real GPS signals at 5 different locations (see Table 5) in the state of Florida, USA. It is clear that we only got *bit sync*, and *NAV* messages for GPS-deprived locations (Crestview, Defuniak Springs, Freeport), whereas we got intermittent *Position fix* in *weak-GPS* areas (Shalimar) and a constant *Position fix* in *strong-GPS* locations (Niceville).
- *Spoofed GPS Dataset:* We collected the same number of I/Q samples from the GPS spoofer, spoofing the same 5 locations, in an uncontrolled lab environment (see Table 5). It is evident from the table that the spoofed data size was much smaller than that of the real data. We found that this was due to there being much less induced noise in the spoofed data than the real one. This in turn was due to the spoofer being much nearer (one millionth) to the GPS receiver compared to the actual GPS satellites. During the spoofed data collection, we always got all the three types of GPS messages, as the spoofer was simulating a *strong-GPS* environment.



(a) Real (b) Spoofed (c) Multi-angular

FIGURE 20. Plots of 4 million (I, Q) value Pairs for different types of dataset from Shalimar.

It is to be noted that though ionospheric and tropospheric errors were included in the collected real datasets, characterization of these are beyond the scope of this work. We are also not explicitly using Doppler Shifts since we are directly using the raw I/Q signal data as input to our system. As a result, we do not need to extract the carrier or phase information of the signal for our experiments.

D. DEEP DIVE INTO DATA

We plot approximately 4 million I/Q value pairs from both the real and spoofed datasets for Shalimar and the corresponding multi-angular projection is shown in Fig. 20. It is clear from Fig. 20 that it is impossible for the human eye to differentiate between the real and spoofed data by simply looking at the signal samples or their plots. We notice similar results for the GPS I/Q data from other locations as well. We must also point out that the set of visible GPS satellites at a given location changes by the time of day at which an observation is made. As a result, the observed GPS data at a given location inherently contains the ''hardware signatures'' of the transmitters on the visible satellites.

As mentioned in Section I, in the current implementation of the GPS system, the unique identity of a GPS satellite is encoded by the transmitted pseudo-random (PRNs) codes (civilian ones) within the broadcast signal. The PRNs obtained at the different data collection locations are presented in Table 6. We refer to the combination of PRNs from the visible satellites at a given location as a *constellation*. Due to the same reason we use the terms *location* and *constellation* interchangeably. It is to be noted that the constellations marked as C_1, C_2, C_3, C_4, C_5 are unique from each other though C_3 is included as a subset of C_4 and C_5. Note that due to this containment relationship it would be interesting to observe whether the proposed CNN is able to differentiate between these three constellations as the underlying features for C_3 are difficult to separate from those of C_4 and C_5.

E. SPATIAL CORRELATION IN THE DATASETS

In order to learn the intrinsic properties of GPS signal data via convolutions, GANSAT needs to exploit the spatial correlations present in the data. To do this we use the idea of the multi-angular projection from Section III-A, to project the received I/Q value pairs at a receiver to a higher dimensional space. Note that for our choice of projection directions, for a single I/Q value the transformed data is given by $\mathcal{X}_{angle}$: $[[(I\cos\frac{i\pi}{8} + Q\sin\frac{i\pi}{8}); i = 1, 2, \cdots, 8]$. In order to study the spatial dependency between the I/Q values, we calculate the correlation between the multi-angular projections of two consecutive I/Q value pairs using both Pearson's and Spearman's correlation coefficients. This is done in order to present a robust perspective as to the efficacy of this transformation.

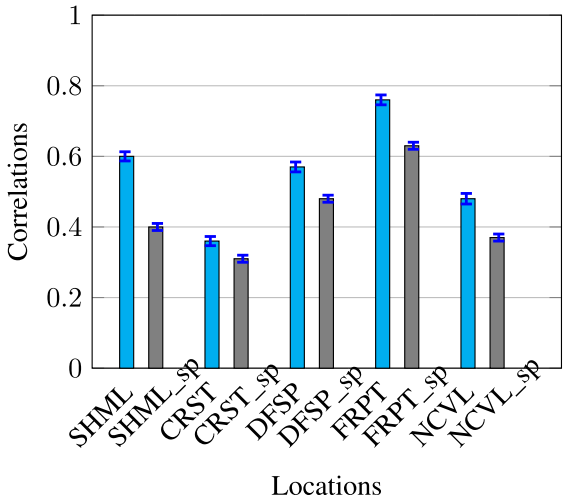


FIGURE 21. Spatial correlation in the datasets in Pearson's method.

The Pearson's correlation at all the 5 locations, including, for both real and spoofed data, are shown in Fig. 21. The Spearman's correlation for the same locations is shown in Fig. 22. Note that the real data for each location is represented in blue whereas the spoofed data is shown in gray. First it must be noted that we typically observe lesser correlation in the spoofed data than in the real ones. CNNs work by capturing the correlation (local features) present in the data and then use the same for classification (or regression). Here, since the nature of the correlations are different for each of the locations in the dataset, this difference can be leveraged to learn *discriminative* features to disambiguate between the various locations. Hence we can conclude that CNNs should be effective for the problem of location estimation using the satellite constellation signatures. This observation is later corroborated via the results from our testbed implementation (Section V).

F. MACHINE LEARNING LIBRARIES USED

Several libraries and tools are available for programming deep neural networks in order to reduce the burden of programming traditional GPUs. We use *Keras* [27] as the frontend with *Tensorflow* [37] as the backend in our implementations. We also use *Numpy*, *Scipy* and *Matplotlib* Python libraries.

G. PERFORMANCE METRICS

To establish the efficacy of any NN-based solution, ''classification accuracy'' is used as the typical performance metric.

TABLE 5. The details of collected datasets.

Location	Real Data		Spoofed Data		Duration
	Size	Messages	Size	Messages	
Shalimar	9.6 GB	*bit sync, NAV, POS*	6.0 GB	*bit sync, NAV, POS*	5 mins
Crestview	9.8 GB	*bit sync, NAV*	6.0 GB	*bit sync, NAV, POS*	5 mins
Defuniak Springs	9.7 GB	*bit sync, NAV*	5.0 GB	*bit sync, NAV, POS*	5 mins
Freeport	9.7 GB	*bit sync, NAV*	5.0 GB	*bit sync, NAV, POS*	5 mins
Niceville	10.9 GB	*bit sync, NAV, POS*	5.0 GB	*bit sync, NAV, POS*	5 mins

TABLE 6. The GPS satellite constellation information.

Locations	Const. Names	Visible Satellite PRNs
Shalimar, FL	C_1	(1, 3, 7, 10, 12, 14, 16, 17, 18, 21, 22, 26, 29, 31, 32)
Crestview, FL	C_2	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 23, 27, 28, 30)
Defuniak Spring, FL	C_3	(1, 9, 10, 11, 12, 13, 14, 15, 16)
Freeport, FL	C_4	(1, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23)
Niceville, FL	C_5	(1, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18)

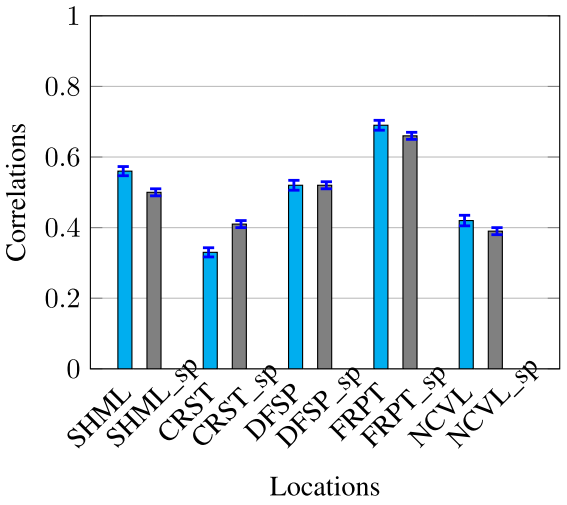


FIGURE 22. Spatial correlation in the datasets in Spearman's method.

However, "classification accuracy" can sometimes be misleading and incomplete when the data is skewed. In such situations, "precision" or "recall" can be used to establish the effectiveness of the model. In the case of two class classification, "precision" of a model is the proportion of positive identifications that are actually correct whereas "recall" measures the proportion of actual positives that were identified correctly. These definitions can be easily extended to the multi-class setting. A confusion matrix uses precision and recall to show how the classification model performs with respect to erroneous predictions (false alarms) and correct classifications. The confusion matrix provides more insights on the performance of a learning model by identifying not only the number of errors, but more importantly the types of errors. Similarly the F_1 score which is the harmonic mean of the "precision" and "recall," is also another widely used performance metric for NN-based classification. As a result, both the confusion matrix and F_1 score are shown and analyzed for our experiments.

V. EXPERIMENTAL RESULTS

This section presents the results of our experiments with the GANSAT framework on real and spoofed raw GPS signal data for the tasks of GPS spoof detection and location estimation using constellation fingerprints. We conducted experiments on real world GPS data obtained from a spoofer and real GPS satellites, All the experiments were conducted on on a Ryzen 8 Core system with 64 GB RAM and a GTX 1080 Ti GPU unit with 11 GB graphics memory. Our experiments consisted of a offline training phase and an online evaluation phase. During the offline training phase, the GANSAT framework was trained to learn the signatures of the real GPS signals as well as the constellation signatures. During the online evaluation phase the trained models were deployed on a system and presented with real or spoofed raw GPS data. Note that the trained model can be deployed on a scaled down version of the hardware used for the experiments. For example the trained models can potentially be deployed on a low end laptop or a portable computer like an Intel Nuk or even a smart phone running the Android OS.

The GANSAT networks are trained on both real and fake raw GPS data collected from $k = 5$ different locations as described before. For the sake of robustness and statistical significance, we present the results from each of the proposed models after averaging over several runs. For describing the results the total number of training examples is denoted by N. Each training sample consists of $n = sample\ size$ I/Q values which correspond to a time t. Note that the signal data obtained is a time series but we do not exploit the temporal variations in the data for this work, rather focusing on the spatial correlations for both the tasks of spoof detection and location inference.

A. SAMPLE SIZE AND NUMBER OF SAMPLES PER CONSTELLATION

Since the GANSAT implementation depends on exploiting the spatial correlation between the I/Q samples, the

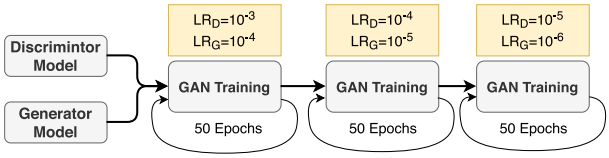


FIGURE 23. The overview of proposed GAN training.

sample size which we have denoted by n plays an important role in our implementations. We performed various sets of experiments with different values of *sample size* to determine the optimum value that captures the spatial correlation in the GPS data. The results of these experiments are presented in Section V-C. Based on these experiments, we decided to use a *sample size* of 1024 and the # of training samples per location (for both real and spoofed data) was fixed at 4000 to avoid the data skewness problem. Any changes in these two parameters are explicitly noted going forward.

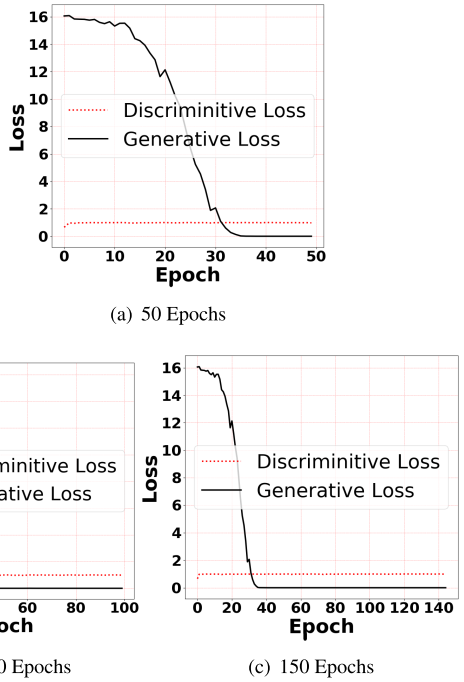
B. PERFORMANCE ANALYSIS FOR SPOOFED SIGNAL IDENTIFICATION

For the first set of experiments, we train the GANSAT model described in section III-B on the collected real raw GPS data and generated raw fake GPS data obtained from the generator. Note that when we train the discriminator without GAN, we notice that it is able to detect `spoofed` GPS signals with 49.22% accuracy, which is similar to what one would obtain with random guessing.

1) TRAINING PHASE

We train both the generator and discriminator through iterative sequential learning to strengthen the generative and discriminative model over time. We perform *categorical cross-entropy* training with *Adam* [25] optimizer for *gradient based optimization*. We use 70%, 10%, and 20% of the total data for training, cross-validation and testing respectively. Note that we pre-process all the data, training or otherwise, using the multi-angular projection as described before.

During the training phase, the generator ($\mathcal{G}$) generates fake GPS I/Q data from a prior constrained random seed using implicit feature information about the *real GPS data* obtained through the discriminator feedback loop. The discriminator ($\mathcal{D}$) is trained using the raw *real* and *generated* GPS data from $\mathcal{G}$. Since GAN training is known to be hard, we experimented with several paradigms for training the GAN models in order to get the best possible performance. This led us to train both the models using a 3-step paradigm which results in faster yet accurate convergence. The 3-step training procedure through 150 epochs is presented in Fig. 23, where $LR_{\mathcal{G}}$ and $LR_{\mathcal{D}}$ represents learning rates of the generator and discriminator respectively. It must be noted that in each epoch the learning rate of the generator is lower than that of the discriminator. This is because we want the generator to be able to generate realistic samples by learning a good approximation to $p_g(g)$, the latent data distribution of *real GPS data*. However we add more layers to the discriminator so that it is able to accurately learn the decision boundary in order to eventually overpower the generator.



(a) 50 Epochs

(b) 100 Epochs

(c) 150 Epochs

FIGURE 24. Generative and discriminative loss of proposed GAN model.

It is clear from Fig. 24 that the generator's loss starts to decrease and the discriminator's loss starts to increase at the beginning of the training. However, after a certain number of epochs both the losses saturate. The generator loss fluctuates more and reaches saturation later (at around 40 epochs) while the discriminator's behavior is more or less stable throughout all epochs. Once both the models are trained, GANSAT is ready for the deployment (online) phase.

2) DEPLOYMENT PHASE

During the deployment phase, we randomly choose GPS signal data from *real GPS satellites*, *a GPS spoofer*, and *fake data from the generator*. We pre-process the *spoofed GPS dataset* to ensure that we have the same sample size and sample space as the *real GPS dataset*, that was collected at different locations in Florida. This was done to ensure that the spoofed data was indistinguishable from the real data using simple features that can be easily learned by a regular discriminator (not trained using a GAN). Using this data we observe $\sim$ 50% spoof-detection rate prior to GAN training. However after the discriminator is trained in the GAN training loop and has learned the data distribution of the real and fake GPS signals, we obtain an improved detection rate. Succinctly we get an accuracy of 99.5% with the spoofed data being generated from either the trained generator or an actual hardware spoofer built by us using software defined radios as described before.

As mentioned earlier, the GAN training and testing were conducted with different values of the sample size and

TABLE 7. Performance of GANSAT for different GAN training settings.

Setting	*sample size*	# samples per const	Total Samples	Before GAN Training	After GAN Training
S_1	512	4000	20000	49.73%	95.6%
S_2	512	8000	40000	50.83%	95.5%
S_3	512	16000	80000	49.88%	96.5%
S_4	1024	2000	10000	49.51%	99.5%
S_5	1024	4000	20000	50.15%	99.5%
S_6	1024	8000	40000	49.86%	99.5%
S_7	2048	2000	10000	53.76%	99.3%
S_8	2048	4000	20000	49.82%	99.5%
S_9	2048	8000	40000	49.83%	99.5%

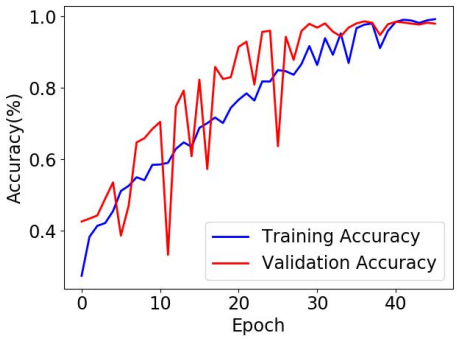
number of training samples per location. The results of these experiments are presented in Table 7. It is evident that the proposed GAN training works reasonably well with different values of both these parameters per location (or constellation).

C. PERFORMANCE ANALYSIS FOR LOCATION PREDICTION

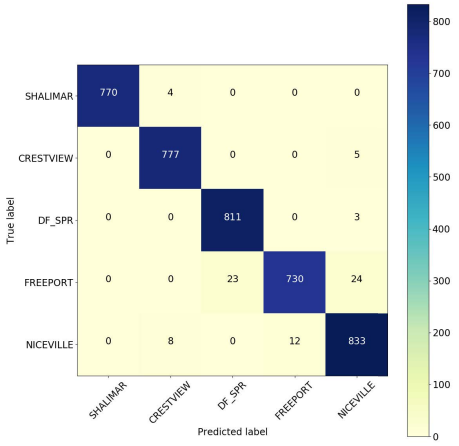
Once the spoofed signals are filtered-out, GANSAT estimates the location using the *constellation* fingerprint of the *real GPS signals* using the proposed 1D-CNN discussed in Section III-C. We obtained 97.82% accuracy, 97.8% precision, 97.82% recall and a F_1-score of 0.98 using GANSAT with the default experimental settings. The accuracy plot and confusion matrices are presented in Fig.25, using 70%, 10%, and 20% of the `real` *GPS data* for training, cross-validation and testing respectively. We note that in general both the training and validation accuracy increases with the number of epochs. However during training, in some epochs the validation accuracy drops suddenly. This is most likely due to degradation of the quality of the parameter estimates due to the optimization algorithm coming up with worse solutions in a given epoch compared to the previous epochs. This is likely due to the inherent complexity of the solution space which is highly non-convex. Note however that the validation accuracy gradually increased over the long run.

It is evident from the confusion matrix that the number of false positives and true negatives are almost negligible. Intuitively the results show that the convolutional filters that were used with the network were able to identify and encode discriminative features for classification. We call these features, which intuitively encode the spatial correlation among the satellite signal data, the "satellite constellation fingerprint" or simply "satellite fingerprint". Also notice that, as pointed out before, the constellations C_3, C_4, and C_5 are quite similar (Table 6), though not the same. However the proposed CNN was able to differentiate between these three constellations even though most of the false positives and true negatives in the confusion matrix (Fig. 25) are corresponding to these three constellations.

We also trained and tested the proposed CNN model with different values of the sample size and number of training samples per location. These results are presented in Table 8. It is noticeable that the proposed architecture works better by increasing the sample size and number of training



(a) Accuracy



(b) Confusion Matrix

FIGURE 25. Accuracy and confusion matrix of the proposed 1D-CNN for satellite constellation classification with *sample size* = 1024, and # samples/location = 4000.

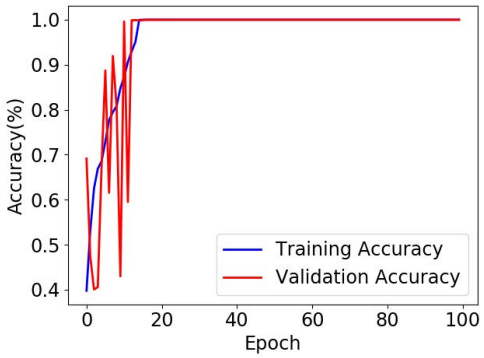
samples per constellation. This observation is intuitively clear, given the fact that a larger sample size coupled with more training samples would reduce the degrees of freedom of the network parameters. However we did observe that the gain in accuracy after reaching a threshold is not enough to justify the increased computational costs that are incurred with such increased sizes. Thus, since the performance did not improve much with a change in sample size from 1024 to 2048, though the training time increases significantly, we decided that a sample size of 1024 with 4000 samples per constellation is sufficient. Furthermore, we also present an ablation study showing the effectiveness of the proposed multi-angular projection. The results are shown in Table 8. It is evident that not using the multi-angular projection results in ∼20% degradation in accuracy for the task of location estimation.

D. PERFORMANCE ANALYSIS FOR IMPROVED LOCATION PREDICTION

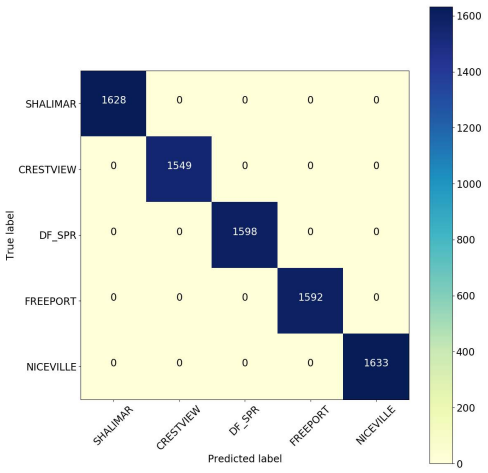
It is interesting to observe that with the implementation of the 2D-CNN model (discussed in Section III-D) for improving the performance of the location prediction algorithm, we get

TABLE 8. Performance of GANSAT for location predictions with variable file parameters.

Sample Size	#samples per const	Without Multi-Angular Projection				With Multi-Angular Projection				Training Time
		Acc	Precision	Recall	F_1-Score	Acc	Precision	Recall	F_1-Score	
512	4000	73.45%	74.34%	74.98%	**0.74**	94.02%	94.05%	94.88%	**0.94**	73 mins
512	8000	74.34%	74.74%	74.43%	**0.74**	95.04%	95.09%	95.04%	**0.95**	90 mins
512	16000	75.35%	75.74%	75.88%	**0.75**	95.68%	95.71%	95.64%	**0.96**	252 mins
1024	2000	76.79%	76.48%	76.29%	**0.76**	97.4%	97.45%	97.45%	**0.97**	523 mins
1024	4000	77.59%	77.28%	77.94%	**0.77**	97.82%	97.8%	97.82%	**0.98**	475 mins
1024	8000	78.49%	78.20%	78.47%	**0.78**	98.42%	98.45%	98.38%	**0.98**	990 mins
2048	2000	74.25%	74.93%	74.03%	**0.74**	93.25%	97.42%	60.70%	**0.75**	702 mins
2048	4000	76.28%	76.97%	76.48%	**0.76**	97.38%	97.38%	97.38%	**0.97**	>1000 mins
2048	8000	77.59%	78.48%	77.45%	**0.78**	98.3%	98.4%	98.6%	**0.98**	>1000 mins



(a) Accuracy



(b) Confusion Matrix

FIGURE 26. Accuracy and confusion matrix of the proposed 2D-CNN for improved location prediction with *sample size* = 2048, and # samples/location = 8000.

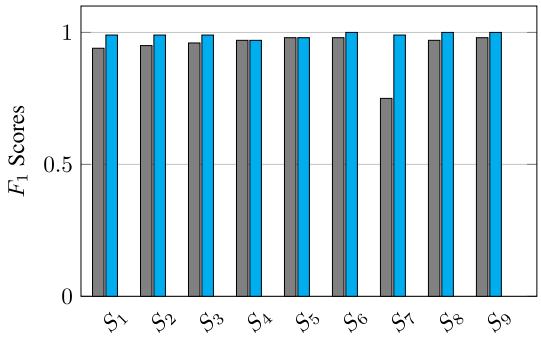


FIGURE 27. Comparison of F_1 score between proposed location prediction schemes. Gray bars: results from the 1D-CNN implementation, blue bars: results from the 2D-CNN implementation.

almost 100% accuracy, 100% precision, 100% recall, and a F_1-score of 1.00 with *sample size* of 2048. It achieves competitive accuracy for other settings as well. The accuracy plot and confusion matrices are presented in Fig.26. We use the same percentages of data for training, cross-validation and testing as with the 1D-CNN. We perform *categorical cross-entropy* training for *gradient based optimization*.

It is clear from the confusion matrix that there are no false positives or false negatives during the testing phase. We also present two comparative bar charts for comparing the F_1 score and training time between the 1D-CNN (basic version) and 2D-CNN (improved version) implementations in Fig. 27 and 28 respectively with different settings. The gray bars represent the results from the 1D-CNN implementations whereas the blue ones represent the ones from the 2D-CNN one. It is noticeable from the charts that the improved version performs significantly better than the basic version with respect to both credibility and time. Moreover, we have noticed a significant impact from using the multi-angular projection for the 2D-CNN implementation which is presented in Table 9. These observations clearly establish that the dimension enhancement during the multi-angular projection positively impacts the representations that can be learned by the network for location estimation.

E. PERFORMANCE ANALYSIS FOR SPOOFED DATA DETECTION

A key element of GANSAT is its ability to detect spoofed GPS signals and filter them out before location estimation. To that effect we use the trained discriminator to differentiate between the collected *spoofed GPS signals* and the *real GPS*

TABLE 9. Performance of GANSAT for improved location predictions with variable file parameters.

Sample Size	#samples per const	Without Multi-Angular Projection				With Multi-Angular Projection				Training Time
		Acc	Precision	Recall	F_1-Score	Acc	Precision	Recall	F_1-Score	
512	4000	76.35%	76.46%	76.93%	**0.76**	97.97%	98.22%	97.75%	**0.98**	34 mins
512	8000	77.29%	77.37%	77.92%	**0.77**	98.67%	98.72%	98.67%	**0.99**	84 mins
512	16000	77.48%	77.93%	77.28%	**0.77**	98.61%	98.65%	98.57%	**0.99**	85 mins
1024	2000	76.93%	76.27%	76.94%	**0.76**	96.85%	98.34%	95.10%	**0.97**	21 mins
1024	4000	77.48%	77.02%	77.48%	**0.77**	99.15%	99.25%	99.12%	**0.98**	43 mins
1024	8000	78.29%	78.49%	78.57%	**0.78**	99.7%	99.74%	99.66%	**1.00**	86 mins
2048	2000	77.58%	77.83%	77.29%	**0.77**	98.60%	99.09%	98.30%	**0.99**	68 mins
2048	4000	78.38%	78.40%	78.74%	**0.78**	99.55%	99.65%	99.42%	**1.00**	174 mins
2048	8000	78.59%	78.47%	78.04%	**0.78**	84%	100%	100%	**1.00**	181 mins

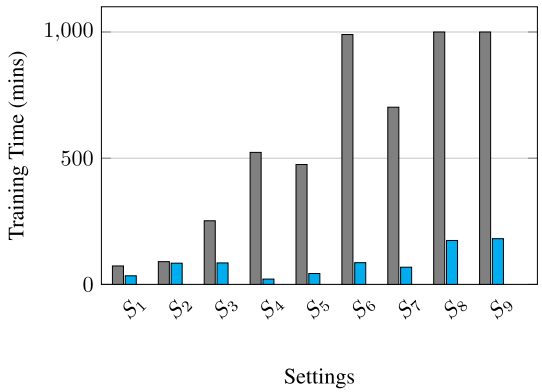


FIGURE 28. Comparison training times between proposed location prediction schemes. Gray bars: results from the 1D-CNN implementation, blue bars: results from the 2D-CNN implementation.

signals at the different locations used for our experiments. Note that the spoofed data can be used by any GPS receiver to estimate the location using positioning and navigation information contained in the data. However, in this work we wanted to see how our proposed CNN models for location estimation with satellite constellation signatures performed when it came to the spoofed data. Succinctly, we wanted to see whether the CNNs could discern satellite constellation signatures from the spoofed GPS data. Intuitively, since the data is generated by a single transmitter - as opposed to a set of satellite based emitters, the features obtained from the spoofed signals should not have enough spatial correlation information to be exploited by the CNNs. As a result one would expect the CNNs to perform poorly with the spoofed data for location inference. This would also establish the fact that the signals from the GPS satellites do contain information that can be exploited to differentiate among different locations.

It was very interesting to observe that the proposed CNN models were able to predict the location from the *spoofed GPS signals* with ∼30% accuracy which is akin to random guess or worse and this proves our intuition empirically. Note that all the spoofed data was transmitted by the *same* SDR and contained information that could be interpreted by any GPS receiver to get location estimates, which are the same as the locations used to collect the real GPS data. Thus if the proposed CNNs simply used the same inherent information from the raw GPS data as used by a GPS receiver, then they would predict the same locations as the GPS receiver. Since this is not the case, it establishes the fact that the CNNs use intrinsic features of the signals received from the satellites which contain discriminating information for location estimation. These signatures from multiple satellites interact to create a ''satellite constellation signature'' which can be exploited for signal classification by location. This also establishes the fact that these features can be used to discriminate between real and fake GPS data. We should also point out that the GANSAT system works with the raw GPS I/Q data and hence its performance does not depend on the received signal strength (RSS) at the receiver as long as the SNR is not below the noise threshold. Thus, the GANSAT system works in environments where the received GPS signal is very weak as opposed to most of the existing systems which would fail to work in weak-signal environments.

F. COMPARISON WITH STATE-OF-THE-ART SPOOF-DETECTION SCHEMES

In this section we present a comparative study of the proposed GANSAT approach for ''GPS spoof-detection using raw signal data'' against some ''state-of-the-art'' techniques for GPS spoof-detection (see Table 10). It must be pointed out that we did not find any ML based methods for robust GPS spoof-detection. Hence the existing techniques that we have considered here are not exhaustive and were only tested with a single run of the experiment. In this setting, it is clear that the GANSAT approach is superior to any existing GPS spoof-detection technique as it achieves nearly 100% accuracy for multiple instances and uses only raw GPS signals. However, due to lack of relevant published literature, we could not give any comparison for the task of location prediction using ''constellation signatures.''

In summary, contributions of GANSAT are as follows:

1) Proposing, implementing and demonstrating the usefulness of multi-angular projection for enhancing the spatial correlation of raw GPS signal data for various raw signal based estimation tasks.
2) Demonstrating the use of spatial correlation from the raw GPS signal data enhanced through multi-angular projection, which can be exploited by GAN and CNN

TABLE 10. Comparison of the proposed approaches with state-of-the-art.

Techniques	# Exp Data	Acc	Comments
Receiver-Autonomous Spoofing Detection [12]	1	100%	Angle of arrival
Psedudo GPS Signal Detection [16]	1	100%	Doppler Shift
INS Monitor [15]	1	100%	Inertial navigation system
Automatic Gain Control Technique [13]	1	100%	GPS Signal Strength
GAN (Ours)	40000	99.5%	Raw GPS Signal

models for spoof detection and location estimation, respectively;

3) Ability to distinguish between *real* and *spoofed* raw GPS signals either in *GPS-deprived*, *weak-GPS* or *strong-GPS* environments;
4) Implementing a system for achieving 99.5% accuracy for spoof detection using the raw GPS signals only. The working principle of GAN does not depend on the received signal strength at the receiver, which is a dominant factor in most of the relevant existing works;
5) Employing a 1D-CNN model that predicts the correct location with 97.82% accuracy by exploiting the spatial correlation as a discriminative attribute for GPS *satellite constellation fingerprinting*;
6) Deploying a 2D-CNN model that estimates the true location with ∼100% accuracy and with much lesser training time; and finally
7) Design of an end-to-end robust system for GPS spoof-detection and location estimation, especially in *GPS-deprived* and *weak-GPS* environments.

G. COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity of the proposed implementations can provide a comprehensive look at the operating landscape of the GANSAT framework. We calculate the complexities for the training phase only, as the trained model provides the output within constant time ($O(1)$) during the deployment phase. Understanding the time complexity of training different types of NN is still an evolving research area. In [38], the authors proved that a NN of depth δ can be trained in $poly(s^{2^\delta})$ time, where s is the dimension of the input and $poly(.)$ denotes a polynomial in the arguments. Note that if δ is constant then for a given dimension s, $poly(s^{2^\delta})$ is a constant and hence such networks are efficiently learnable. Note that the convolution operations of CNNs add additional time complexity along with the forward and back-propagation operations. In [29], the authors noted that the time complexity for training all the convolutional layers is: $O(\sum_{\tau=1}^{\zeta}(\eta_{\tau-1}\nu_\tau^2.\eta_\tau\rho_\tau^2)$, where ζ is the number of convolutional layers, τ is the index of a convolutional layer, $\eta_{\tau-1}$ is the number of input channels for the τ^{th} layer, ν_τ is the spatial size of the filters at the τ^{th} layer, η_τ is the number of filters at the τ^{th} layer and ρ_τ is the size of the output features of the τ^{th} layer. In the GANSAT 1D-CNN model, there are 4 convolutional layers and 5 fully connected layers and hence this adds an additional time complexity for training those four convolutional layers. Similarly, an additional time complexity for the single convolution layer is used in the 2D-CNN implementation.

TABLE 11. Time complexities for training of GAN, 1D-CNN, and 2D-CNN implementations.

Classes	Models	Complexity	# Hyper-params
[*Real, Spoofed*]	GAN	$poly(0.8 \times N_{data}^{2^8})$	10.5 Million
$[C_1, C_2, C_3, C_4, C_5]$	1D-CNN	$poly(0.8 \times N_{data}^{2^5}) \times O(\sum_{\tau=1}^{4}(\eta_{\tau-1}.\nu_\tau^2.\eta_\tau.\rho_\tau^2) + poly(0.8 \times N_{data}^{2^4})$	27.5 Million
$[C_1, C_2, C_3, C_4, C_5]$	2D-CNN	$poly(0.8 \times N_{data}^{2^8}) \times O(\sum_{\tau=1}^{1}(\eta_{\tau-1}.\nu_\tau^2.\eta_\tau.\rho_\tau^2) + poly(0.8 \times N_{data}^{2^2})$	6.3 Million

The time complexities for the two parts of GANSAT implementation are presented in Table 11. The proposed GAN, 1D-CNN and 2D-CNN implementations had 8, 9 and 3 layers of neurons respectively. Note that we have used different sample sizes and the total number of samples (for training, validation and testing). As mentioned before, 80% of data is used for training and validation purpose. For example, the complexity of GAN with 8 layers using 80% of data (N_{data}) samples for training and validation, is $poly(0.8 \times N_{data}^{2^8})$. On another note, we should also mention that for quick estimates of the training time, the number of hyper-parameters can be used as a ball-park number for the complexity. Thus, if there are more hyper-parameters, then the training time is proportionately larger. We present the number of hyper-parameters in Table 11 with a sample size of 1024.

1) LIMITATIONS OF GANSAT

Finally, before concluding we must point out that the main limitation of GANSAT is its dependency on the training data the model was trained on, like any other data driven solution. In this regard, for GANSAT to successfully estimate the location from raw GPS signal data, it has to be trained with a wide variety of GPS constellations, even for the same location (since the visible constellation at a location changes with time). We validate GANSAT on collected data from 5 different locations, hence, our testing framework is limited to the GPS signals from those 5 locations only. However, since the locations were randomly selected, our implementation and results establish the possibility of using such a system for large scale deployments with approximate models using transfer learning over temporal models. We have not explored this direction in this work and leave that to a future study.

VI. CONCLUSION

This paper presents the GANSAT framework, which provides robust defense against GPS spoofers. The GANSAT system and associated algorithms achieve 99.5% accuracy for the task of GPS spoofer detection. The proposed approach is resilient to any kind of spoofing attacks, as it works on the physical layer of the devices. The framework is also able to predict the approximate location of a software GPS receiver with ∼100% accuracy, using the raw GPS signals, leveraging the concept of "satellite constellation fingerprinting" in GPS-denied and weak-GPS environments. The robustness of the proposed approach is established using a testbed implementation and the observed efficacy is explained using local characteristics of GPS signal data as represented by spatial correlations. Additionally, time complexity of the implementation is compared with other GPS spoofer detection schemes.

Though the GANSAT framework is tested on data from 5 locations, the success of this implementation proves the feasibility of applying AI/ML based methods for large-scale location prediction in GPS denied environments and leads to the possibility of incorporating the same in civilian and commercial GPS systems. Going forward we would like to study the feasibility of large scale deployment of such systems using the idea of temporal transfer learning for adapting temporally stale GANSAT models to work with real time data using minimal retaining. This would open up the possibility of large scale deployments of such models which might then require industry level collaboration for implementation and testing.

ACKNOWLEDGMENT

The views and conclusion contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Air Force.

REFERENCES

[1] S. Starker, "The application of spread spectrum techniques in satellite orbit determination and navigation systems," in *Localization and Orientation in Biology and Engineering*. Cham, Switzerland: Springer, 1984, pp. 28–31.
[2] D. Roy, T. Mukherjee, and M. Chatterjee, "Machine learning in adversarial RF environments," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 82–87, May 2019.
[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[5] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Cham, Switzerland: Springer, 2006.
[6] D. Roy, T. Mukherjee, M. Chatterjee, and E. Pasiliao, "Detection of rogue RF transmitters using generative adversarial nets," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.
[7] M. L. Psiaki, T. E. Humphreys, and B. Stauffer, "Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies," *IEEE Spectr.*, vol. 53, no. 8, pp. 26–53, Aug. 2016.
[8] D. Fudenberg, F. Drew, D. K. Levine, and D. K. Levine, *The Theory of Learning in Games*, vol. 2. Cambridge, MA, USA: MIT Press, 1998.
[9] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Proc. Engineering Applications of Neural Networks*, 2016, pp. 213–226.
[10] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, Sep. 2018.
[11] T. Humphreys, B. Ledvina, M. Psiaki, B. O'Hanlon, and P. Kintner, "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," in *Proc. 21st Int. Tech. Meeting Satell. Division Inst. Navigat.*, vol. 2, 2008, pp. 1198–1209.
[12] P. Montgomery, T. Humphreys, and B. Ledvina, "Receiver-autonomous spoofing detection: Experimental results of a multi-antenna receiver defense against a portable civil GPS spoofer," Tech. Rep., 2009, pp. 124–130.
[13] D. M. Akos, "Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC)," *J. Inst. Navigat.*, vol. 59, no. 4, pp. 281–290, 2012.
[14] K. C. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang, "A practical GPS location spoofing attack in road navigation scenario," in *Proc. 18th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2017, pp. 85–90.
[15] Ç. Tanil, S. Khanafseh, M. Joerger, and B. Pervan, "An INS monitor to detect GNSS spoofers capable of tracking vehicle position," *IEEE Trans. Aerosp. Electron Syst.*, vol. 54, no. 1, pp. 131–143, Feb. 2018.
[16] H. Wang, W. Cheng, C. Xu, M. Zhang, and L. Hu, "Method for identifying pseudo GPS signal based on radio frequency fingerprint," in *Proc. 10th Int. Conf. Commun., Circuits Syst. (ICCCAS)*, Dec. 2018, pp. 354–358.
[17] J. Randall, O. Amft, J. Bohn, and M. Burri, "LuxTrace: Indoor positioning using building illumination," *Pers. Ubiquitous Comput.*, vol. 11, no. 6, pp. 417–428, Aug. 2007.
[18] M. Luimula, K. Sääskilahti, T. Partala, S. Pieskä, and J. Alaspää, "Remote navigation of a mobile robot in an RFID-augmented environment," *Pers. Ubiquitous Comput.*, vol. 14, no. 2, pp. 125–136, Feb. 2010.
[19] M. Ficco, F. Palmieri, and A. Castiglione, "Hybrid indoor and outdoor location services for new generation mobile terminals," *Pers. Ubiquitous Comput.*, vol. 18, no. 2, pp. 271–285, Feb. 2014.
[20] V. Perekadan, T. Mukherjee, C. Banerjee, and E. Pasiliao, "RF-MSiP: Radio frequency multi-source indoor positioning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5259–5268.
[21] Y. Wei, W. Li, and T. Chen, "Node localization algorithm for wireless sensor networks using compressive sensing theory," *Pers. Ubiquitous Comput.*, vol. 20, no. 5, pp. 809–819, Oct. 2016.
[22] F. Yang, S. Li, H. Zhang, Y. Niu, C. Qian, and Z. Yang, "LIPO: Indoor position and orientation estimation via superposed reflected light," in *Springer Personal and Ubiquitous Computing*, 2019.
[23] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. Boca Raton, FL, USA: CRC Press, 2017.
[24] P. Hanson and T. Holden, "Digital spread spectrum GPS navigation receiver," U.S. Patent 5 943 363, Aug. 24, 1999.
[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[27] F. Chollet. (2015). *Keras: The Python Deep Learning Library*. Accessed: Jun. 30, 2020. [Online]. Available: https://keras.io
[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[29] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5353–5360.
[30] Ettus Research. (2018). *USRP B210*. Accessed: Jun. 30, 2020. [Online]. Available: https://www.ettus.com/product/details/UB210-KIT
[31] CTTC. (2019). *GNSS-SDR*. Accessed: Jun. 30, 2020. [Online]. Available: https://gnss-sdr.org
[32] CTTC. (2020). *GNSS-SDR: Configurations*. Accessed: Jun. 30, 2020. [Online]. Available: https://gnss-sdr.org/conf/
[33] Takuji Ebinuma. (2018). *GPS-SDR-SIM*. Accessed: Jun. 30, 2020. [Online]. Available: https://github.com/osqzss/gps-sdr-sim
[34] W. Gurtner. (1997). *RINEX: The Receiver Independent Exchange Format Version 2*. Accessed: Jun. 30, 2020. [Online]. Available: https://www.ngs.noaa.gov/CORS/RINEX-2.txt
[35] NUAD. (2019). *Blade RF*. Accessed: Jun. 30, 2020. [Online]. Available: https://www.nuand.com/product/bladerf-xa4/

[36] C. Limited. (2018). *GPS Test, Version 1.5.8*. Accessed: Jun. 30, 2020. [Online]. Available: https://play.google.com/store/apps/details?id=com.chartcross.gpstestplu%s&hl=en_US
[37] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.
[38] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 855–863.

DEBASHRI ROY received the Ph.D. degree in computer science from the University of Central Florida, USA, in 2020. She was working as a research scientist at University of Central Florida while conducting experiments for this paper. She is currently an Associate Research Scientist at Northeastern University. Her research interests include multimodal data fusion, radio frequency machine learning, beam selection, and channel estimation in mmWave band and terahertz communication.

TATHAGATA MUKHERJEE received the M.S. and Ph.D. degrees in computer science from Florida State University. He is currently an Assistant Professor of computer science with The University of Alabama in Huntsville, Huntsville. Prior to his current position, he was the Chief Scientist at Intelligent Robotics Inc., a non-profit DoD research laboratory. His research interests include cyber security, adversarial machine learning, cognitive radio networks, optimization, and graph theory.

ALEC RIDEN is currently the bachelor's degree in computer science with The University of Alabama in Huntsville. His research interests include cyber security, machine learning, reverse engineering, and malware and exploit research.

JARED PAQUET is currently an Engineer at the University of Florida Research and Engineering Education Facility. His research interests include network hardware and software.

EDUARDO PASILIAO received the B.S. degree in mechanical engineering from Columbia University, New York City, NY, USA, in 1992, and the M.S. degree in coastal and oceanographic engineering and the Ph.D. degree in industrial and systems engineering from the University of Florida, Gainesville, FL, USA, in 1995 and 2003, respectively. He is currently a Senior Research Engineer with the Air Force Research Laboratory. He has coauthored over 100 peer-reviewed journal and conference publications. His research interests include graph theory, combinatorial optimization, discrete mathematics, network science, and machine learning. He has been a member of the American Institute for Aeronautics and Astronautics and the Institute for Operations Research and Management Sciences.

ERIK BLASCH (Fellow, IEEE) received the B.S. degree in mechanical engineering from MIT, in 1992, the master's degrees in mechanical engineering, in health science, and in industrial engineering (human factors) from Georgia Tech, in 1994, 1995, and 1995, and the M.B.A., M.S.E.E., M.S.Econ., and Ph.D. degrees in electrical engineering from Wright State University, in 1998, 1998, 1999, and 1999. He was a graduate of the Air War College, in 2008. In 1996, he served in active duty with the United States Air Force. He is currently a Principal Scientist at the U.S. Air Force Research Laboratory (AFRL) in the Information Directorate at Rome, NY, USA. He is also a Program Manager for the AFOSR DDDAS Program.