DeepSIM: GPS Spoofing Detection on UAVs using Satellite Imagery Matching

Nian Xue
nian.xue@nyu.edu
New York University
New York, USA

Liang Niu
liang.niu@nyu.edu
New York University
New York, USA

Xianbin Hong
xianbin.hong@liverpool.ac.uk
University of Liverpool
Liverpool, UK

Zhen Li
lizh0019@gmail.com
Nanjing University of Aeronautics and Astronautics, Nanjing, China
Shanghai Grandhonor Infotech Co.Ltd

Larissa Hoffaeller
larissa.hoffaeller@student.hpi.de
Hasso Plattner Institute
Potsdam, Germany

Christina Pöpper
christina.poepper@nyu.edu
New York University Abu Dhabi
Abu Dhabi, UAE

ABSTRACT

Unmanned Aerial Vehicles (UAVs), better known as drones, have significantly advanced fields such as aerial surveillance, military reconnaissance, cadastral surveying, disaster monitoring, and delivery services. However, UAVs rely on civilian (unauthenticated) GPS for navigation which can be trivially spoofed.

In this paper, we present *DeepSIM*, a satellite imagery matching approach to detect GPS spoofing attacks against UAVs based on deep learning. We make use of the camera(s) a typical UAV is equipped with, and present a system that compares historical satellite images of its GPS-based position (spaceborne photography) with real-time aerial images from its cameras (airborne imagery). Historical images are taken from, e. g., Google Earth or NASA WorldWind. To detect GPS spoofing attacks, we investigate different deep neural network models that compare the real-time camera images with the historical satellite images. To train and test the models, we have constructed the *SatUAV* dataset (consisting of 967 image pairs), partially by using real UAVs such as the DJI Phantom 4 Advanced. Real-world experimental results show that our best model has a success rate of about 95% in detecting GPS spoofing attacks within less than 100 milliseconds. Our approach does not require any modification of the existing GPS infrastructures and relies only on public satellite imagery, making it a practical solution for many everyday scenarios.

CCS CONCEPTS

• **Security and privacy** → **Mobile and wireless security**; • **Information systems** → *Global positioning systems*; • **Computer systems organization** → Embedded and cyber-physical systems.

KEYWORDS

GPS spoofing detection, UAV, deep learning, neural networks

ACM Reference Format:
Nian Xue, Liang Niu, Xianbin Hong, Zhen Li, Larissa Hoffaeller, and Christina Pöpper. 2020. DeepSIM: GPS Spoofing Detection on UAVs using Satellite Imagery Matching. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3427228.3427254

1 INTRODUCTION

In recent years, there has been growing interest in UAVs that are increasingly used in many scenarios [63]. As high-performance, low-cost, and intelligent UAVs have become more affordable and attainable, UAVs have experienced a quick transition from the military to the civilian domain. For example, the FBI has started to utilize UAVs for both public surveillance and military reconnaissance [55]. A second example is that the International Telecommunication Union (ITU) is considering a UAV-aided 5G wireless communication framework [51]. Additionally, after the devastating earthquake and the following disastrous tsunami in Fukushima, Japan in 2011, a Honeywell T-Hawk UAV equipped with special radiation sensors was used for investigating the damaged reactor, where humans could not approach [49].

Although deploying applications based on UAVs has strong benefits, unfortunately, a few accompanying risks have gradually appeared [40]. Media reports with respect to cyber-attacks on UAVs have become common. For instance, in 2012, a rotor-based UAV, Camcopter S-100, whose GPS signal was blocked by an unknown actor, then crashed into a ground control van, killed an engineer and injured two remote pilots [28]. Moreover, our work is further motivated by a real-life spoofing attack: the well-known Iran-U.S. RQ-170 incident, where an American UAV was captured by Iranian forces near the city of Kashmar in 2011 due to jamming satellite signals, followed by a GPS spoofing attack [53].

In this paper, we study the problem of detecting GPS spoofing attacks for the specific context of UAVs. Our approach is to compare the difference between aerial photos taken by camera-enabled UAVs and pre-existing satellite images as a form of out-of-band location verification. Both a GPS module and a visual sensor such as a camera are usually outfitted in a UAV system. Attackers hardly affect the visual information using traditional GPS spoofing antennas. By and large, it is also very difficult for an attacker to carry out attacks for vision by changing the geographic features in the real world. As a consequence, visual equipment could be used as an alternative reference for GPS spoofing attack detection due to its independence. As a proof of concept, we present *DeepSIM*, a GPS spoofing detection approach for UAVs via satellite imagery matching; the system design of DeepSIM is shown in Figure 1. We also constructed the *SatUAV* image dataset for training and testing our matching models.

For the spoofing detection process, we first collect a reference dataset by letting the target UAV take an aerial photograph at its real geographic location from the air and send it to the ground controller. After receiving the photo, the ground controller acquires the relevant satellite imagery assembled in the UAV systems or from public resources, i. e., Google Earth or NASA WorldWind. By determining a threshold for the similarity between the real photos taken by the UAV's camera and the satellite images of a GPS position that the UAV claimed, our system can detect GPS spoofing attacks by identifying whether two images were taken at the same location.

This approach contains a number of challenges: First, images taken by UAVs differ substantially in resolution, rotation, quality, and other features (such as brightness and saturation) compared with map imagery. Second, satellite imagery can be very different to corresponding aerial photos due to external factors such as weather, people, vehicles, light, and seasonal changes. Third, datasets with paired images taken by UAVs are not readily available and creating such a dataset (i. e., SatUAV) for training our deep learning models is challenging, since in many countries security restrictions and drone regulations limit drone operations. As an example, getting official U.S. approval for extended drone operations that would be needed for mapping activities would very likely require authorization from the Federal Aviation Administration (FAA) as a prerequisite. Accordingly, laws and policies further exacerbate the difficulty of obtaining raw airborne imagery.

In order to best tackle the first two challenges, we make use of the power of deep learning methods. Specifically, Convolutional Neural Networks (CNNs) are especially powerful for image recognition, classification, and segmentation: As early as 2016, it was reported that Artificial Intelligence (AI) had outperformed a human in a top-5 error rate (3.57% vs. 5.1%) on ImageNet's Large Scale Visual Recognition Challenge (ILSVRC) [6]. Benefiting from rapid development of GPU computing and CNN's powerful learning and feature extraction ability, the use of CNNs promises accurate yet efficient detection. To the best of our knowledge, this paper is the first attempt to detect GPS spoofing attacks on a single camera-enabled UAV based on satellite vs. aerial pairs via deep-learning methods.

Advantages. Compared with other conventional detection methods of GPS spoofing attacks, our countermeasure is a software-only and non-invasive approach requiring no modifications of existing GPS signals, GPS satellites, or GPS-capable receivers in practice. Additionally, our approach does not introduce redundant receivers or extra specific and expensive hardware for the analysis of the GPS signal characteristics with acceptable power consumption. Due to the development of AIoT (AI + IoT), some UAVs have already been equipped with neural computing accelerator modules (e. g., the DJI Spark Drone has an Intel chip that enables deep learning features [19]); such tendencies will potentially make our proposed method even more efficient in the near future. Even a single UAV is able to independently detect GPS spoofing attacks without cooperation from other GPS-capable devices. Finally, our approach can be run in two ways: (1) on the remote controller of a ground station or (2) on the on-board computer (OBC) of a UAV.

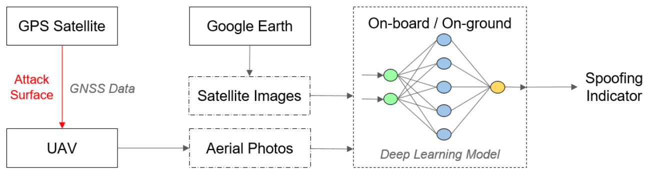


Figure 1: The system design of DeepSIM.

Contributions. Our main contributions are as follows:

- We build *SatUAV*, a pioneering dataset of 967 satellite-aerial image pairs. We use it to train our neural networks.
- We propose DeepSIM, a GPS spoofing-detection approach for UAVs via satellite imagery matching. We propose four different models based on CNN image-matching algorithms.
- We construct a prototype DeepSIM system and evaluate its performance with the proposed four models. We further identify the best two models for practical deployment based on experimental results, and measure their power consumption.
- We describe the use of DeepSIM in two ways of operation: DeepSIM can either be run by a central controller after receiving the imagery from the UAVs (on-ground) or can be executed directly on the UAV itself (on-board). We demonstrate the feasibility and accuracy of our concept by experiments using real-world data.

All source codes and datasets are available at our GitHub project website[1].

2 BACKGROUND AND PRELIMINARIES

In this section, we start by outlining basic knowledge of GPS, GPS attacks, and countermeasures, followed by a description of Neural Networks as context for the introduction to our proposed *DeepSIM* technique.

2.1 Global Positioning System

The U.S. Global Positioning System (GPS) has been extensively used in various scenarios in the last decades. In the core GPS system, medium Earth orbit satellites continuously broadcast navigation signals in six different orbital planes. Devices equipped with GPS-capable receivers can compute their 3D position and local time by measuring the time of arrival (ToA) of at least four satellite signals. However, GPS does not provide integrity and authenticity protection for civilian signals, making attacks on the system possible.

2.1.1 Attacks on GPS. GPS signals can be divided into two categories: civilian signals and military signals. Compared to military GPS signals that use a secret military code to improve the anti-jamming and anti-spoofing properties, civilian GPS signals are neither encrypted nor authenticated. That is to say, civilian GPS

[1] https://github.com/wangxiaodiu/DeepSim

is inherently fragile and susceptible to GPS attacks (i. e., jamming and spoofing attacks). The susceptibility of GPS to attacks has been investigated since the year 2001 [39, 61, 62, 65].

GPS Jamming Attacks. GPS jamming attacks block GPS signals or interfere with the target victims in order to keep them from receiving legitimate GPS signals. One dangerous impact of GPS jamming attacks is that they can prevent the systems that rely on GPS signals from being able to "navigate" to their destinations. Studies regarding jamming attacks can be found in [7, 15, 21].

GPS Spoofing Attacks. On the other hand, GPS spoofing attacks attempt to fabricate similar and fake, but more powerful satellite signals to deceive GPS-capable receivers. As a consequence, a victim will lock onto the spoofing signal rather than the legitimate GPS signal. Several successful spoofing experiments were carried out [16, 17, 23, 61, 64], indicating that such attacks are no longer theoretical assumptions. Coupled with the rise of programmable radio platforms such as HackRF [43], the costs of commercial off-the-shelf spoofing devices go down drastically, rendering GPS spoofing attacks the *de facto* pressing threat.

2.1.2 Defenses against GPS Attacks. Given the lack of security of civilian GPS signals, a large number of countermeasures were proposed to mitigate the potential risks. These countermeasures can be roughly classified into *prevention* and *detection* approaches.

GPS Attack Prevention. GPS attack prevention methods using cryptographic techniques have been broadly presented and discussed [13, 29, 50, 66], which is similar to the solution employed by military GPS signals. Nevertheless, such cryptographic methods either demand to upgrade the existing GPS infrastructure [48] or modify the GPS signal structure. Also, the proposed key distribution mechanism is challenging. Hence, we conclude such countermeasures based on modifications of current GPS infrastructure including receivers, emitters and associated GPS devices are unlikely to be implemented in the near future. Moreover, according to [44], it is impossible to avoid replaying attacks using merely encryption.

GPS Attack Detection. In comparison to prevention methods, the GPS attack detection methods attract more attention from academia and industry. Previously proposed countermeasures are summarized in [22] and [46]. Further, these countermeasures can be roughly categorized into the following three categories: (*i*) detection at signal level, (*ii*) direction of arrival sensing, and (*iii*) out-of-band techniques.

- **Detection at Signal Level.** GPS spoofing detection at signal level tries to identify the abnormal signal based on the physical features of signals. For example, spoofing checks based on physical signal waveform can be found in [1, 35, 46, 48]. The authors of [25] used multiple co-located GPS receivers and leverage spatial noise correlation to detect spoofing signals. Although there is no modification on existing GPS signal structure, these techniques do require modification of receivers and GPS-associated devices or need special devices for the analysis of signal features. In addition, such methods significantly increase the costs and complexity of deployment of countermeasures.
- **Detection at Direction of Arrival Sensing.** Another class of detection approaches utilize the direction of arrival sensing. For example, Montgomery et al. [37] distinguished spoofing signals by measuring the angle of arrival. A method based on range-only information to detect GPS spoofing in platoons of vehicles was introduced by Swaszek et al. [60]. Jansen et al. [24] proposed Crowd-GPS-Sec to detect and localize GPS spoofing attacks on moving airplanes.
- **Detection Using Out-of-band Techniques.** GPS spoofing attacks can also be detected by comparing the GPS position information with alternative sources of location. Examples of out-of-band means for positioning that can be combined with GPS are visual sensors, Inertial Measurement Units (IMUs), WiFi, altimeters, enhanced long-range navigation (E-LORAN), and cellular-based location [2, 42]. Our proposed approach falls into this category. Methods with similar auxiliary equipment include [10, 33, 47, 68], which will be discussed and compared with our method in Section 7 (Related Work).

2.2 Deep Neural Networks

Deep learning models, especially Deep Neural Networks (DNN), are currently the most popular machine learning technique, widely used in various areas. Here, we shortly introduce the neural networks used in DeepSIM.

Residual Neural Networks: CNNs [32] have triggered a series of successes in the domain of image classification. However, vanilla CNNs are susceptible to vanishing gradients. At the moment, Residual Network (ResNet) [12] stands out from a variety of different Neural Networks. ResNet, a milestone in the history of CNN images, utilizes skip connection to propagate information over layers. By doing this, the network is able to understand global features and resolve the problem of degrading accuracy. Hence, ResNet enables scientists and researchers to build deeper networks. It has been proven in [12] that training this kind of network is much easier than training vanilla deep convolutional neural networks. Taking advantage of its powerful ability to extract features from images, we utilize ResNet as part of the backbone network in our models.

SqueezeNet: SqueezeNet [18] released in 2016 is a small and compact CNN architecture with fewer parameters and layers. It could replace the ResNet for much less memory usage and faster inference speed with compromised feature extraction ability. SqueezeNet has less layers and uses smaller convolutional kernels, which leads to much less parameters and faster inference speed. Thus, SqueezeNet can be more easily run on devices with limited memory and computing power such as a Raspberry Pi and a smartphone.

Siamese and Semi-Siamese Networks: Siamese Network [5] is an artificial neural network originally designed to recognize human faces. Generally, it contains two or more identical subnetworks, having the same architecture with the same weights and parameters. Siamese Network performs well on the task of finding similarities or relationships between two comparable input vectors, which inspires us to use it to determine whether two images (i. e., the satellite image and corresponding aerial photo) are paired.

Semi-Siamese Network [67] is a variant of the original Siamese Network. Instead of fully weight-shared networks, Semi-Siamese Network only shares weight between CNN backbones and then distinguishes their outputs by a learnable Fully Connected Network rather than Euclidean distance. Since only CNN backbones of the whole network share weights, the model is called Semi-Siamese Network.

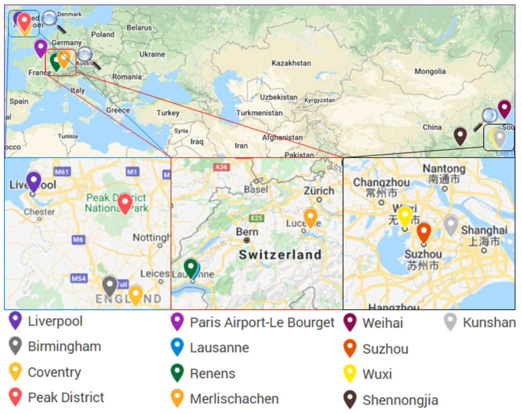


Figure 2: Areas where aerial photos are collected.

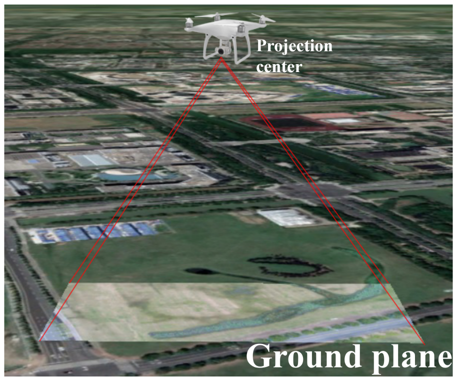


Figure 3: An example of central projection in our scenario.

3 SATUAV DATASET CONSTRUCTION

To validate our approach that we will present in detail in Section 4, we require a collection of aerial photography and satellite imagery. Only after such a dataset is constructed, we can use it to train our neural networks. Then the neural networks check whether an aerial photo matches a satellite image or not. However, one current challenge is that there is no existing dataset of paired aerial photography versus satellite imagery in the *status quo*, and thus we first have to build such an image dataset by collecting appropriate images ourselves. The dataset we constructed is named *SatUAV* that includes a large amount of paired satellite imagery and aerial photography from 13 cities or regions around the world, see Figure 2.

Images in our dataset are part of two categories: aerial photography and satellite imagery. As of now, the total number of image pairs in our dataset is 967 (appr. 12.08 Gigabyte).

3.1 Aerial Photography

To collect satisfactory aerial photos for training and testing, we took a huge amount of new photos by ourselves using a real UAV, namely a DJI Phantom 4 Advanced. We also leveraged existing aerial photos from the senseFly website[2] to enrich our dataset.

In total, we gathered 967 aerial photos. Among them are 605 realistic scene photos with a flight height of 120 m (393.7 ft) that were captured using our own UAV; 343 of these photos were taken in Suzhou, China, and 20 photos were captured in Kunshan, China. The rest were gathered in Weihai, Shennongjia and Wuxi respectively. To test the generalization ability of our models, we additionally collected 107 photos from 4 areas in the UK The camera used for shooting is DJI FC6310. To get sufficient details of the ground, we set the original pixel resolution to 5472 × 3078.

To make our dataset abundant and diverse, we obtained aerial photos from different sources with disparate terrain features. To this end, we included 362 photos available on the Internet from the senseFly website. The collected photos can be divided into four groups in terms of four different cities. Three of the cities are in Switzerland and the fourth is Le Bourget Airport near Paris, France. All of the aerial photos were taken by eBee drones. The pixel resolutions from the Internet are 4608 × 3456 and 5472 × 3648. The first data group with 160 photos was taken in a small Swiss village of Merlischachen by a Canon IXUS 125 HS with a flight height of 162 m (531.4 ft). The second group with 40 photos was captured by a Parrot Sequoia camera in outskirts of Renens, a municipality in Switzerland. The flight height was set to 100 m (328.1 ft). The third group with 113 photos came from Lausanne, Switzerland. These photos were captured by a senseFly S.O.D.A. camera, and the flight height is also about 100 m (328.1 ft). The last 49 images taken by a Canon IXUS 125 HS camera display the aerial view of Paris Le Bourget Airport at the height of 120 m (393.7 ft). As for the shooting time period, the photos from Merlischachen were generated in April 2013; the second group of images was taken in October 2016; the third image collection (Lausanne) was captured in January 2000; the last airport aerial images were obtained in June 2013. Summaries of specification regarding aerial photography are listed in Table 1.

3.2 Satellite Imagery

Satellite imagery, initially for military purpose, is now attainable and freely available to the public. For our dataset we downloaded satellite imagery from Google Earth[3] containing a large collection of Earth observation imagery. In spite of satellite imagery resolution ranging from 15 meters of resolution to 15 centimeters, we used the highest resolution we can get from Google Earth. The pixel resolutions of the satellite imagery are set to three categories—3840 × 2160, 4800 × 3200, and 4800 × 3600, which match the ratio of the aerial photography respectively. Satellite imagery regarding Merlischachen was taken by a satellite in June 2015; spaceborne photography corresponding to Renens was generated in March 2018; the third group of imagery with respect to Lausanne was collected by satellite in March 2015 and August 2016 separately; the last 30 airport images were generated in May 2018. The satellite imagery associated with photos captured by our own UAV were taken in July 2017 (Suzhou). And Shennongjia's images were taken in November 2017 and January 2018. All the satellite photos of Kunshan Weihai and Wuxi were taken in 2018. Moreover, all the UK data were captured from 2018 to 2019. Table 2 gives an overview of the specifications regarding the used sets of satellite imagery.

[2]https://www.sensefly.com/education/datasets/

[3]https://www.google.com/earth/

Table 1: Specification summary of aerial photography.

Place	Pixel resolution	Ratio	#Images	Flight height	Shooting time	Scenario features	Camera	Usage
Suzhou	5472×3078	16:9	343	120 m	9/2018–3/2019	lakeside city	DJI	training&test
Kunshan	5472×3078	16:9	20	120 m	10/2018	heritage town	DJI	training&test
Weihai	5472×3078	16:9	57	120 m	10–11/2018	coastal city	DJI	training&test
Shennongjia	5472×3078	16:9	9	120 m	12/2018	mountain forests	DJI	training&test
Wuxi	5472×3078	16:9	69	120 m	3/2019	downtown	DJI	training&test
Birmingham	5472×3078	16:9	37	120 m	4/2019	city park	DJI	test-only
Coventry	5472×3078	16:9	15	120 m	5/2019	university campus	DJI	test-only
Liverpool	5472×3078	16:9	41	120 m	4/2019	urban&park	DJI	test-only
Peak District	5472×3078	16:9	14	120 m	5/2019	national park	DJI	test-only
Merlischachen	4608×3456	4:3	160	162 m	4/2013	lakeside village	Canon IXUS	training&test
Renens	4608×3456	4:3	40	162 m	10/2016	cropland	Sequoia	training&test
Lausanne	5472×3648	3:2	113	100 m	1/2000	industrial zone	S.O.D.A.	training&test
Le Bourget Airport	4608×3456	4:3	49	120 m	6/2013	airport	Canon IXUS	training&test

Table 2: Specification summary of satellite imagery.

Place	Pixel resolution	Ratio	#Images	Shooting time
Suzhou	3840×2160	16:9	343	7/2017
Kunshan	3840×2160	16:9	20	3/2018
Weihai	3840×2160	16:9	57	4-5/2018
Shennongjia	3840×2160	16:9	69	11/2017, 1/2018
Wuxi	3840×2160	16:9	9	4/2018
Birmingham	3840×2160	16:9	37	5/2019
Coventry	3840×2160	16:9	15	5/2019
Liverpool	3840×2160	16:9	41	3/2018
Peak District	3840×2160	16:9	14	6/2018
Merlischachen	4800×3600	4:3	160	6/2015
Renens	4800×3600	4:3	40	3/2018
Lausanne	4800×3200	3:2	113	3/2015, 8/2016
Le Bourget Airport	4800×3600	4:3	49	6/2013

3.3 Image Pairing Calibration

To create the image pairs, our basic principle is to keep the coverage and location of the satellite imagery consistent with those of the aerial photography.

In general, UAVs collect lots of data during flight missions, including image data, position and orientation system (POS) data and metadata. Thus, we decided to adapt a central projection method to calculate location and coverage from POS data included in aerial photos. By doing this, the four vertex coordinates of the projection rectangle on the ground (see Fig. 3) can be determined. Then, we use these coordinates (geographic data) to obtain corresponding satellite images from Google Earth. Finally, we use the images to train our neural network models to recognize spoofing attacks.

We next introduce *central projection*, which is the projection from one plane onto another plane from a central point (also known as a projection center); however, the projection center is not on either projection plane. In our case, a focal point inside the physical camera is the projection center. Both aerial photos and satellite imagery can be considered as projections from this point. To simplify the model, suppose that the plane (i. e., the image sensor plane) where aerial photos lie is parallel to the plane of the satellite imagery on the ground.

The vertical slices of aerial photography projection on the image sensor plane and ground plane and the covered area on the ground are shown in Figures 4a–4c. Notations used in these figures and in the description of our algorithm are summarized in Table 3.

Combining central projection with Figure 4a, we can deduce the camera's horizontal α and vertical β angle of views, where F is the effective focal length, and W_0 and H_0 represent the width and height of the film, respectively:

$$\begin{cases} \alpha = \arctan \frac{W_0}{2F} \\ \beta = \arctan \frac{H_0}{2F}. \end{cases} \tag{1}$$

In fact, the real coordinate is affected by a UAV's yaw ϕ, pitch θ and roll ψ degrees. Therefore, the projection direction onto the ground is not perfectly perpendicular (i. e., the red rectangle in Figure 4c). In practice, there is a slight shift (i. e., the black rectangle in Figure 4c) due to some factors such as wind and mechanical errors. Thus, we need to introduce some amendment. Considering Figure 4b, Figure 4c and UAV's pitch θ and roll ψ degrees, we can deduce the following results:

$$\begin{cases} x_a = (L - L_0) \tan\theta \sin\phi \\ y_a = (L - L_0) \tan\theta \cos\phi \\ x_b = (L - L_0) \tan\psi \cos\phi \\ y_b = (L - L_0) \tan\psi \sin\phi \\ x_0 = x + x_a + x_b \\ y_0 = y + y_a - y_b. \end{cases} \tag{2}$$

Coupling the central projection and Figure 4b, we have derived the following equations from Equations (1) and (2):

$$\begin{cases} W_1 = (L - L_0) \tan(\alpha - \psi) \\ W_2 = (L - L_0) \tan(\alpha + \psi) \\ H_1 = (L - L_0) \tan(\beta - \theta) \\ H_2 = (L - L_0) \tan(\beta + \theta), \end{cases} \tag{3}$$

Table 3: Summary of notations

Term	Definition	Term	Definition
F	Camera's focal length	W_0	Width of the aerial photo
L	Flight height above sea level	H_0	Height of the aerial photo
L_0	Altitude of geographic location	(x_0, y_0)	LAT/LONG of central point
ϕ	Yaw degree of the UAV	(x, y)	LAT/LONG of shooting point
θ	Pitch degree of the UAV	(x_a, y_a)	Amendment of pitch value
ψ	Roll degree of the UAV	(x_b, y_b)	Amendment of roll value
α	Camera's horizontal angle of view	(x_i, y_i)	Coordinate of corresponding corner point of the projected rectangle image, i=1,2,3,4
β	Camera's vertical angle of view		

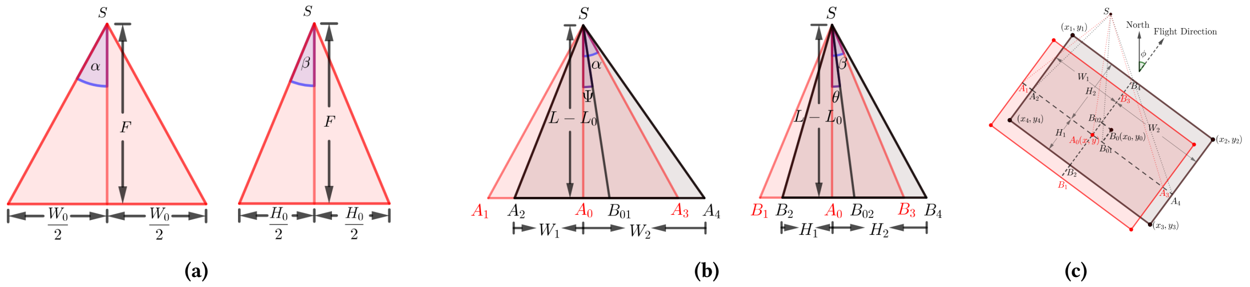

S
α
β
F
$\frac{W_0}{2}$
$\frac{H_0}{2}$
(a)
$L-L_0$
Ψ
θ
A_1 A_2 A_0 B_{01} A_3 A_4
W_1 W_2
B_1 B_2 A_0 B_{02} B_3 B_4
H_1 H_2
(b)
North
Flight Direction
(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_4, y_4)
$B_0(x_0, y_0)$
(c)

Figure 4: (a) Vertical slices of aerial photography projection on the image sensor plane. S is the projection center. The red areas indicate the projection rays are always perpendicular to the image sensor plane. (b) Vertical slices of aerial photography projection on the ground. S is the projection center. The black areas indicate the actual projection with allowable tolerance, and red areas indicate the ideal (perfectly perpendicular) projection. (c) Vertical slices of aerial photography projection on the ground. S is the projection center. The black areas indicate the actual projection with allowable tolerance, and red areas indicate the ideal (perfectly perpendicular) projection.

where W_1 and W_2 denote the shifted distances affected by the roll ψ, while H_1 and H_2 are the shifted distances affected the pitch θ (indicated in the Figure 4b). Then, we can obtain:

$$\begin{cases} x_1 = x + H_2 \sin\phi - W_1 \cos\phi \\ y_1 = y + H_2 \cos\phi + W_1 \sin\phi \\ x_2 = x + H_2 \sin\phi + W_2 \cos\phi \\ y_2 = y + H_2 \cos\phi - W_2 \sin\phi \\ x_3 = x - H_1 \sin\phi + W_2 \cos\phi \\ y_3 = y - H_1 \cos\phi - W_2 \sin\phi \\ x_4 = x - H_1 \sin\phi - W_1 \cos\phi \\ y_4 = y - H_1 \cos\phi + W_1 \sin\phi. \end{cases} \tag{4}$$

The rectangle area on the ground shot by the drone can be calculated through the above equations. It is worth noting that we merely use an approximation method to reduce the complexity. There are two factors that can affect the accuracy of computations. First, when taking pictures, the drone lens needs to shoot photos as downwards (vertically) as possible, and thus the actual angle of the camera should be restricted to close to 90 degrees. Otherwise, the error of the coverage of ground projection becomes very large. Second, ideal projection ground is flat terrain. Incline landscape such as hills will introduce deviation.

Figure 5 shows typical examples of pairs of aerial photography versus satellite imagery from our dataset at different places. According to the above equations, we can conclude that with the increase of flying height, the photos taken by UAVs can cover larger surface areas, thus making the aerial photos contain more visual features, which suggests that photos captured in different places can be more easily distinguished from one another. However, due to rules and inherent limitations of UAVs, the height cannot be infinite, and regulations vary from country to country. For example, in the US, aviation authorities recommend flying below 400 feet / 121 meters above ground level to avoid possible collision with manned aircraft such as airplanes or helicopters.

4 SYSTEM DESIGN OF DEEPSIM

DeepSIM is the integration of UAVs, a ground-based UAV controller (or ground station), a spoofing indicator and communications between UAVs and ground station. To simplify the model, we assume that the camera is mounted directly under the UAV, with the camera's movement being consistent with the UAV's movement.

Underlying Concept. The underlying concept of our proposed GPS spoofing-detection approach for UAVs is inspired by natural features in the real world. Our natural environment has many random features and unpredictable factors, which are very hard for attackers to simulate and manipulate. We design this approach based upon the difficulty of forgeability of the visual features in nature. Such features include, but are not restricted to, roads, rivers, mountains, streets, landscapes, skyscrapers, landmarks, etc.

4.1 System Overview

Each part of the DeepSIM system (Fig. 1) works as follows: **1)** The UAV processes GPS signals from the satellites by a GPS-capable receiver to determine its own position and it takes aerial photos of the ground using a camera. **2)** The UAV controller provides an operation and monitoring platform for the UAV and is also responsible for running models to detect spoofing attacks (unless this is run on the UAV directly). **3)** Communications refer to the channel for remote control and exchange of video and other data between the UAV and the UAV control center. **4)** The spoofing indicator will alert administrators if spoofing attacks are detected.

4.2 Attacker Model

We consider an attacker emitting fake GPS signals to UAVs by injecting, modifying, replaying, or delaying GPS messages. As a result, the target UAV will compute a wrong geolocation and send incorrect location data back to the ground station. This may result in the UAV being displayed at a manipulated position in the map on the remote control panel and it may result in misleading the UAV to take a wrong itinerary.

Moreover, the attacker may block or jam the communication channel between the UAV and the ground station by active signal transmissions on the frequency channel used for communication. This would cause the UAV to lose contact with the ground station.

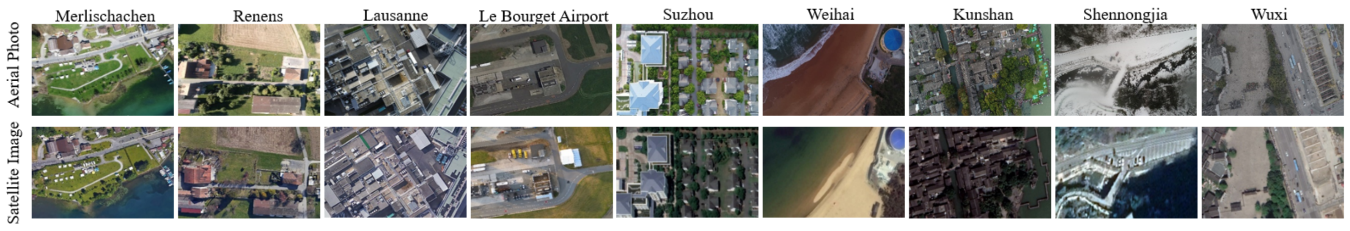


Figure 5: Examples of paired images from our dataset.

While this goes beyond GPS spoofing, we propose an On-board Model of DeepSIM that does not rely on an active and available communication channel between the UAV and the ground station.

4.3 Image Matching for Spoofing Detection

The design of DeepSIM contains two types of detection for GPS spoofing attacks: on-demand and periodical detection. Once the victim UAV is spoofed from its original geolocation to another destination specified by the attacker, the differences between real-time aerial photos and satellite imagery of the spoofed destination are generally obvious. Our system can quickly identify such inconsistencies and then raise warning notifications to administrators.

On-demand Detection. On-demand detection is used for UAVs in sensitive areas where the possibility of GPS spoofing attacks is high, such as battlefronts and protected airspace. The on-demand GPS spoofing detection works as follows: (1) A UAV controller sends a spoofing-detection command to a target UAV through the wireless channel. (2) The UAV adjusts trimming (trimming is often used to maintain straight, level flight and stabilize the drone) and takes environmental photos. (3) The UAV sends aerial photos back to the central controller. (4) The controller receives the photo from the UAV and obtains the corresponding satellite image using the method described in Section 3.3. (5) The controller runs algorithms to compare the aerial photos with the satellite imagery, and then determines whether the image pairs match each other or not. (6) The controller raises an alarm if the two images are unpaired.

Periodical Detection. Periodical detection is applied when UAVs work in low-risk areas. The controller does not need to send spoofing-detection commands to the target UAV while the UAV itself will send back the aerial photos periodically (e. g., every 30 seconds) to the controller for detecting spoofing attacks. Besides, UAVs can perform detection by itself once the communication channel is compromised by attackers. Periodical detection does not put an extra burden on the UAVs, and it is more friendly to battery-powered and computationally limited drones.

4.4 Deep Learning Model

Since there are two input vectors, i. e., satellite images and aerial photos, it is natural to use neural networks with two branches. In addition, as satellite images and aerial images are very similar, sharing weight between two branches is a sensible choice. Based on such intuition, we propose four different deep-learning based models: *Distance Threshold*, *Siamese ResNet*, *Semi-Siamese Network*, and *Siamese SqueezeNet* as candidates to find appropriate neural networks to detect GPS spoofing attacks for different scenarios. For the sake of simplicity, we named them Model 1, Model 2, Model 3 and Model 4 in turn. All proposed models have two weight-sharing branches and compare I_1, the image taken by the UAV camera, and I_2, the image retrieved from the satellite image databases like Google Earth. Algorithm 1 describes the overall idea of our GPS spoofing-detection. Finally, we decide to adopt *Siamese ResNet* for on-ground detection and *Siamese SqueezeNet* for on-board detection based on experimental results described in Table 9. Here, we only give the description of the models we adopt for on-ground detection and on-broad detection. More details regarding the other two models and comparison results and reasons can be found in the Appendix A.

Algorithm 1: Overall Algorithm of Four Models

```
Input: $I_1$, image taken by a UAV, $G$, geographical data
       obtained by the UAV's GPS-capable receiver
Output: Whether the UAV is spoofed or not
Initialization: The model $M$;
Retrieve satellite image $I_2$ according to $G$;
$R = M(I_1, I_2)$;
if $R = 0$ then
    return 'Not spoofed';
end
return 'Spoofed';
```

4.4.1 *On-ground Detection Model: Siamese ResNet.*

On-ground detection utilizes *Siamese Network*, a prevailing two-branch image pairing model, to perform spoofing detection. The structure of the Neural Network model for the detection of GPS spoofing attacks is shown in Figure 6. In our implementation, the trainable network is composed of CNN layers and Fully Connected layers, in which the CNN layers are initialized by a pretrained ResNet to speed up training with a relatively small volume of training data. Since it consists of Siamese Network and ResNet, we call it Siamese ResNet.

During training, the Siamese Network tries to learn the features so that if two images are paired, their Euclidean distance is small. This goal is achieved by optimizing its loss function, Contrastive Loss [8], which is shown below:

$$\mathcal{L} = \frac{1}{2}\{(1-y) \times d^2 + y \times max(margin - d, 0)^2\}, \tag{5}$$

where d denotes the Euclidean distance of two feature maps output from the network, y the label if two images are from the same location, $y = 0$, and $margin$ a hyperparameter set before training.

4.4.2 *On-board Detection Model: Siamese SqueezeNet.*

As mentioned earlier, ResNet is introduced as the backbone for its powerful feature extraction ability in On-ground Model. However, On-ground Model requires more than 2 GB memory to run. As a matter

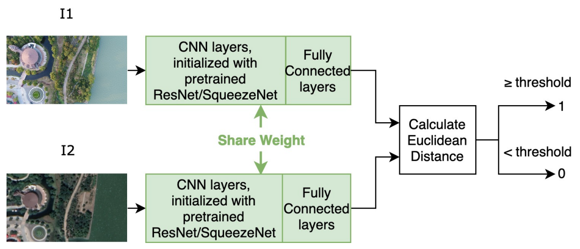

I1
CNN layers, initialized with pretrained ResNet/SqueezeNet
Fully Connected layers
Share Weight
I2
CNN layers, initialized with pretrained ResNet/SqueezeNet
Fully Connected layers
Calculate Euclidean Distance
≥ threshold
1
< threshold
0

Figure 6: Neural Network Structure. The black arrows indicate data flow, the green section is the trainable model. The whole network shares weight. Our On-ground Model uses ResNet as the backbone, while our On-board Model uses SqueezeNet as the backbone.

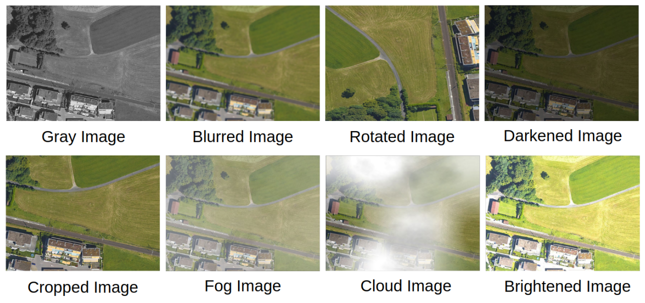


Figure 7: Images after data augmentation.

of fact most UAV on-board computers do not have that much memory. Thus, we adopted SqueezeNet, a lightweight neural network, aiming to enable the UAV to detect GPS spoofing with on-board computation. The major difference of this model and the models mentioned above is the backbone of the model, as shown in Figure 6. Therefore, Siamese Network with SqueezeNet runs about 6 times faster than Siamese Network with ResNet in theory.

In order to prove that this model can be run on a UAV, the related experiments are all conducted on a Raspberry Pi SOC with similar limited computational resources as a UAV computation board. This model can be run on-board on UAVs to enable the UAV to detect GPS spoofing independently when attackers also deploy jamming to disrupt the communication between UAV and ground controller.

5 EXPERIMENTS AND EVALUATION

To demonstrate the effectiveness and efficiency of the proposed methods, all models were tested on the SatUAV dataset.

5.1 Experimental Setup

Hardware Environments. Both On-ground Model and On-board Model were trained and evaluated on a single NVIDIA® Tesla® V100 GPU with 16GB GPU memory. The CPUs on this V100 GPU server are two Intel® Xeon® E5-2683 v3 @ 2.00GHz CPUs. Besides, the On-board Model is also tested on a Raspberry Pi 3B+ whose CPU is Cortex-A53 @ 1.4GHz and memory is 1GB.

Software Environments. The GPU server runs a 64-bit Linux system. Raspberry Pi runs a 32-bit Linux system for ARM. All the experiments are implemented in Python 3 with Pytorch [45] as the deep learning framework. For the V100 GPU server, CUDA and cuDNN [4] are installed to accelerate the Neural Networks training and inference process.

5.2 Implementation Details

Dataset. Prior to SatUAV, no dataset of paired aerial photos versus satellite imagery existed. As a response, we constructed SatUAV and leveraged all the 967 SatUAV image pairs in our experiments.

Preprocessing. To evaluate the proposed On-ground and On-board models, we divided the whole dataset into two parts: the training data and test data. 80% of the dataset (688 pairs) was randomly chosen to become the training data, and the rest of them (172 pairs) became the test data. We will report our results on both training data and test data. All images were resized to 960×720 (4:3) to fit into memory before being fed into neural networks. Higher resolution, of course, contains more details that could be helpful to the neural networks, but higher resolution also leads to higher memory usage and smaller batch size. We compared several image resolutions and decided to choose 960×720 as input.

Data Augmentation. In the real world, a variety of conditions can be encountered, such as different rotation, grayscale, croppings, and blurred images due to bad weather conditions. At the same time, we created a dataset of image pairs that cover only a limited set of conditions and cities. In order to enrich our dataset and make the models robuster, we augment the data by eight different augmentation techniques that simulate numerous variations of landscape conditions: rotation, graying, cropping, blurring, darkening, brightening and several weather conditions like clouds and fog. The image augmentations are generated with the use of the Python computer vision library `OpenCV` [14] and the Python image augmentation library `Imgaug` [26]. We choose parameters determining the strength and amount of rotation, cropping, and blurring for different condition simulations. Figure 7 shows the images after data augmentation. By adding the augmented images, we obtain 7,740 image pairs as input for the training of our models.

Negative Samples Mining. In all the experiments, paired satellite images and aerial photos were considered as positive samples and labeled as 0. However, there were only positive samples (i. e., paired images) in the SatUAV dataset, which required a negative sample mining process to generate enough negative samples for training and testing. Thus, we used random unpaired images from the dataset as negative samples. To ensure there is a similar number of positive and negative samples, we employ a random picking trick, described in Algorithm 2. To fairly compare the result, the number of generated negative samples was the same as the number of positive samples during testing.

Network Architecture. A pretrained ResNet-34 (without the last Fully Connected layer) played the role as the backbone CNN network in On-ground Model. The final pooling layer of this pretrained ResNet-34 is an average pooling layer whose kernel size is 7 and stride is 1, the output feature map size is $24 \times 17 \times 512$. As for On-board Model, a SqueezeNet v1.1 without its last Fully Connected layer played the role as the backbone CNN network, and the output feature map size is $59 \times 44 \times 512$.

Algorithm 2: Random Picking. This algorithm is invoked when selecting a pair of images as the input in each training or testing iteration.

```
Input: $i$, satellite image array $A$, aerial image array $B$
Output: The $i$-th image pair, and its label $l$
$s = size(A)$;
$a = A[i]$;
$l$ = random.pick([1,0]);
if $l = 1$ then
    $shift$ = random.pick($[1, 2, ..., s - 1]$) ;
    $i = (i + shift)\%s$;
end
$b = B[i]$;
return $a, b, l$;
```

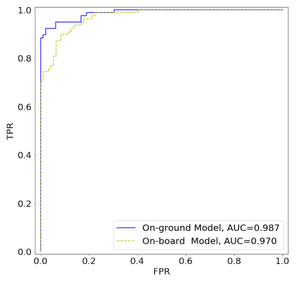
Table 4: Layers after backbone of On-ground Model and On-board Model. *FC(N)* **indicates Fully Connected layer with** *N* **output nodes.** *BN* **indicates Batch Normalization [20].** *DO(p)* **indicates Dropout with** *p* **as possibility [56].** *Conv_1×1(N)* **indicates 1 by 1 convolutional kernel with** *N* **output channels.**

Model	Layers after backbone		
On-ground	FC(2048), BN, ReLU	FC(512), DO(0.2), PReLU	FC(8)
On-board	Conv_1x1(16), FC(512), BN, ReLU	FC(64), DO(0.2), PReLU	FC(8)

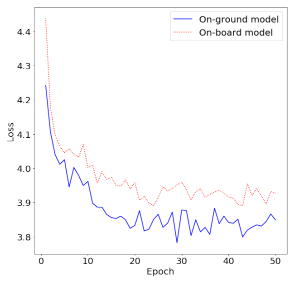
The layers attached after backbones in On-ground Model and On-board Model are described in Table 4. As shown in the table, a 1-by-1 convolutional layer with 16 output channels is applied right after the SqueezeNet backbone for On-board Model. This convolutional layer is introduced to reduce dimensionality of the feature map so that the computational load can be further reduced for on-board execution.

Transfer Learning. In both models, a machine learning technique called Transfer Learning was leveraged. It helped to accelerate the experiments, shorten training time, and make our experiments feasible using a relatively small dataset. Transfer Learning reuses trained models by applying it to new models or fine-tuning them to fit into different but related tasks. There are mainly two kinds of transfer learning for CNN. One uses pretrained CNN as fixed feature extractor, the other is to fine-tune the pretrained CNN. On-ground Model and On-board Model mainly apply the second type of transfer learning. We have also explored the first type Transfer Learning in Model 3, which will be elaborated in the Appendix.

For On-ground Model, a pretrained ResNet-34 was introduced as the backbone and its weights were used to initialize the backbone, whose parameters or weights were updated as the training went on. Similarly, a pretrained SqueezeNet v1.1 was introduced to On-board Model as its backbone and its weights were used to initialize the backbone, whose parameters or weights were updated as the training went on. As Figure 6 shows, the backbones and the layers after backbones were trained together in both models.



(a) ROC curves and AUC scores.



(b) Training loss.

Figure 8: ROC curves and training loss of both models.

Optimization. The loss functions of On-ground Model and On-board Model are both Contrastive Loss as described in Equation 5. Stochastic Gradient Descent with momentum [58] was applied to optimize the models and update the trainable parameters. The learning rate was set to decay by 0.1 every *step* epochs, where *step* is a hyperparameter. We kept tracking and updating the model having the highest accuracy during the whole training process. The final experimental results were based on the saved model with highest accuracy rather than that after the training was completed.

Hyperparameters. We conducted different experiments to explore the optimal hyperparameters for these two models. The final results are reported here. Both models used 3×10^{-4} as the initial learning rate, $step = 10$ and $momentum = 0.9$. They are trained for 100 epochs on original data and 50 epochs on augmented data. Here, an epoch means one full training cycle through the entire training dataset while training a machine learning model. It takes much longer to train on the augmented data so we reduced the number of epochs for training on augmented data. Margin as in Contrastive Loss is set to 4. The batch sizes are 4 for both models.

5.3 Experimental Results

Training Process: Figure 8 demonstrates the loss change during the training process of both models.

Performance: Experiments for two models were conducted on the whole SatUAV dataset. We use the standard F1-score as a main measure to evaluate model performance:

$$\mathcal{F}_1 = 2 \times \frac{precision \times recall}{precision + recall}. \tag{6}$$

We refer to Table 5 for the numeric experimental results including accuracy, precision, recall and the F1 score for each model. Figure 8 (a) shows their Receiver Operating Characteristic (ROC) curves and their Area Under the Curve (AUC) scores. We further tested both models using data augmentation technique. The results are shown in Table 6. We can see that both accuracy and F1 Score benefit from this technique.

Time Complexity and Power Consumption: On-ground Model was trained on our GPU server, for about 20 minutes per epoch on augmented data and for about 2.4 min per epoch on original data. On-board Model was also trained on the GPU server, for about 10.2 minutes per epoch on augmented data and for about 1.5 minutes on original data. For testing on our GPU server, it merely took less

Table 5: Experimental results of the On-ground Model and On-board Model, trained without data augmentation.

Model	Dataset	Threshold	Accuracy	Precision	Recall	Error Rate	F1 Score	Memory Usage	Disk Usage	Power Consumption
On-ground	SatUAV-Training	2	0.869	0.837	0.911	0.131	0.873	2.9 GB (CPU)	1.7 GB	N/A
(Model 2)	SatUAV-Test	2	**0.895**	0.855	0.954	**0.105**	**0.902**	4.5 GB (GPU)		
On-board	SatUAV-Training	2	0.849	0.782	0.971	0.151	0.866	285 MB (CPU)	85 MB	1.1 Watts
(Model 4)	SatUAV-Test	2	**0.826**	0.770	0.917	**0.174**	**0.837**			

Table 6: Results after data augmentation, using SatUAV-Test for testing and 2 as threshold. With data augmentation, not only both accuracy and F1 score of two models increased, but also the error rate for On-ground Model reduced 50% and the error rate for On-board Model lowered 37% respectively.

Model	Accuracy	Precision	Recall	Error	F1 Score
On-ground	**0.948**	0.930	0.979	**0.052**	**0.954**
On-board	**0.890**	0.871	0.936	**0.110**	**0.903**

Table 7: Average running time (left) and attack probability by different deviations (right) of the On-ground and On-board models. The results indicate the estimated probabilities of spoofing attacks become higher as the deviations increase.

Model	Time (sec/pair)	Hardware	15m	30m	45m
On-ground	< 0.1	Ground GPU server	0	0.15	0.20
On-board	10.48	Raspberry Pi	0	0.05	0.15

than 0.1 sec/pair to predict for both models. Besides, the On-board Model took about 10.5 sec/pair for testing on a Raspberry Pi 3B+. The average running time of all models is summarized in Table 7.

For the On-board Model, power consumption is also an important factor since overburden may affect the mission time for a UAV with limited battery capacity. We use a USB tester to measure the power of Raspberry Pi. The power consumption with/without running the model are 3.23 watts and 2.13 watts respectively. Given that the running time is short, see Table 7, and GPS spoofing detection is not always being performed, it does not reduce battery life largely.

Error Tolerance: In order to further study the error tolerance of our method, we generated 5 groups of satellite images which are shifted from the original position towards the four cardinal directions — north, south, east, and west. The deviations are set to 15, 30 and 45 meters respectively, since the standard accuracy of GPS is about 15 meters (49 feet) [36]. In total, 60 images were fed into the On-ground and On-board Models. The results are also shown in Table 7. Larger value means higher probability of spoofing attacks. We can find that as the deviation grows, DeepSIM is also gradually likely to think such deviation is due to GPS attacks, which is consistent with the common intuition as well.

Generalization Ability: The ability to generalize our technique to non-trained geographical areas is important since it is not always possible to train the models on data from the target areas. To study the generalization ability of the proposed models, we collected extra test data from four places in the UK: Birmingham, Coventry, Liverpool, and Peak District. Since our models are trained on non-UK data, we can evaluate the generalization ability based on their accuracy for the UK data. On the 107 pairs of the UK-images, the accuracy for the On-ground Model is 0.888 (F1 score 0.889), for the On-board model 0.841 (F1 score 0.872). Comparing these results to Table 5, we conclude that both On-ground and On-board Models have good generalization ability for unfamiliar places.

6 DISCUSSION AND LIMITATIONS

6.1 Pros and Cons of the Models

Currently a UAV with a visual sensor is the standard equipment. All models just need photos to detect GPS spoofing attacks. Their thresholds are adjustable, and they can easily be combined with other detection methods like time-series analysis, detection at signal level, etc.

The major limitation is caused by the nature of our methods. They highly rely on the diversity of visual features, which makes the models perform badly on landscapes like desert, oceans, woods whose features are monotonous and hard to distinguish.

In addition, natural phenomena can degrade the accuracy. For instance, standard cameras can hardly capture objects in the dark night; fog can blur the view of the landscape; strong light can cause destructive interference in image formation. In such cases, our methods can still be effective if the UAV is equipped with special devices, e. g., a night-vision camera, or uses selected state-of-the-art techniques, e. g., haze removal using dark channel prior [11].

Even though our best model indicates approximately a 95% accuracy, there is a chance that FP (False Positive) and FN (False Negative) predictions will happen. The following methods can be used to reduce the system error:

1) Analyse the result as time sequence. We could detect attacks every 5 seconds. Since it's unlikely for attackers to attack victims in a short period cycle (e. g., every 10 seconds), if the system predicts there is only one or two attacks per minute, it is highly possible they are false alarms. In the real world, given that the possibility of GPS attacks is low, DeepSIM may raise alarms after inconsistencies are detected three times in succession. Thus, the false positive rate also can be reduced down to acceptable levels in practice.

2) Raise an exception to the operators. When the system detects attacks, it raises a spoofing alarm, and shows the aerial photo and the satellite image to the UAV operator, let the operator make final decision whether there is an attack or not.

3) Analyse the sensors fusion results. With the IMU and other sensors as an auxiliary, we can analyse the sensor fusion results and do cross-validation to improve the accuracy and avoid errors.

6.2 Attacks on our Approach

Attacks on the Communication Channel. Our method provides alternative ways to detect GPS spoofing attacks, which is completely different from traditional methods. Generally, an attacker merely equipped with GPS spoofing antenna cannot attack our communication channel for aerial photography transmitting, since our approach utilizes out-of-band technique rather than GPS signals. To both carry out a GPS spoofing attack and compromise the communication channel at the same time will be more difficult than GPS-only attacks. Up to now, there are no reported incidents regarding such attacks, also called Video Replay Attack [28], but it should be noted. In this case, an attacker can feed the control unit with recorded video. One countermeasure to secure communication stream is to use cryptographic techniques. Compared with GPS channel, it is relatively easy to deploy. In addition, to implement attacks on communication channel, it requires the attacker to be near the victims. As a result, it is much easier to find the attacker compared with GPS-only attackers who can be very far from the victims, usually tens of kilometers away. Even if the attackers managed to implement attacks on communication channels, our On-board Model could be launched to detect spoofing without the assistance of remote controller.

Attacks on Our Models. There are two kinds of attacks on Neural Networks: In white box attacks, the attackers have the knowledge of neural network weights and parameters; in black box attacks, the attackers only have the knowledge of the model or no knowledge at all. Depending on the goal, the attacks can be classified into two categories: non-targeted attacks try to make the neural network misclassify, targeted attacks try to make the neural network classify to a certain class. To attack DeepSIM, the attackers need to cover the landscape [30]. Without the knowledge of network parameters, it is very hard to achieve such an attack. Even if the attackers are able to conduct a White Box Attack, they will need to change large areas of landscapes, which is considerably difficult. Moreover, attackers cannot guarantee that the modified camouflages must be captured by UAVs in proper position to manipulate our models.

Attacks on Landscape. Another possible attack to our approach is the landscape attack. Theoretically, our approach is vulnerable to such attacks when a UAV is still hovering just above featureless areas, such as ocean, desert, prairie. However, UAVs are often used in scenarios like urban centers, rural areas, industrial districts, etc. in practice. These scenarios have plenty of topographic features (e. g., skyscrapers, landmarks, rivers, roads, etc.). Our approach is meaningful and feasible in most cases, and it is proved by our experiments. Another conceptual way is to change the landscape or set camouflage that can affect the collected photos. However, for an attacker, this method is unfeasible to carry out in reality. Even if an attacker can modify landscape on earth, the coverage is limited to a narrow zone. Once the victim UAV flies across such camouflaged range, attacks on landscape will become invalid.

6.3 Future Usage of the Database

SatUAV has been made open through our project website (we obtained consent from SenseFly). Researchers in security, computer vision, machine learning can benefit from our open dataset. For computer vision researchers, this dataset can be leveraged in domain adaptation, image matching and retrieval, adversarial attacks on vision models and so on. For GPS security, SatUAV and DeepSIM set up a benchmark and baseline for GPS spoofing detection via computer vision methods. Security researchers can develop insightful findings using SatUAV, like secure vision-based localization.

6.4 Post-Detection Countermeasures

We have assumed that the GPS signals sent by the attacker are transmitted wirelessly and that the UAV computes incorrect geolocations due to the fake GPS signals. Once GPS spoofing attacks are detected, there are mainly three strategies:

1) Fly out of the GPS spoofing area. The spoofed UAV should keep on going towards one direction. Continuously it checks whether the GPS location is spoofed or not until it reaches an area where the spoofing warning is lifted.

2) Land in the current position. The target UAV can use the mature vision-based landing technique [3, 31]. However, if the landing zone is dangerous (e. g., lake, ocean and crater), then it needs to find somewhere safe to land or just hover at a low height until the end of the spoofing attack.

3) Return to home automatically. This is a promising solution, attracting lots of research. Currently, there are two mainstream methods: one is based on GPS information, and the other is based on image matching. However, when GPS signal is spoofed or jammed, GPS-based methods are unavailable. Recently, a post-mission autonomous return method which works well without needs of GPS signal was proposed by Nguyen et al. [38].

7 RELATED WORK

In recent years, research regarding security of UAVs has sparked large interest with the popularity of drones and security concerns about UAVs [9, 27, 28, 54]. Significant efforts have been put into developing GPS spoofing-detection techniques. According to the latest literature review in [28], one of the pressing challenges presented by UAVs is the GPS spoofing attack, which is also the most common attack on UAVs. However, the mainstream approaches are limited to analyzing the GPS signal itself. Among those traditional countermeasures, some impose modifications in the structure of GPS signals or infrastructures by using cryptographic techniques [13, 29, 41, 66], some introduce additional specific equipment to analyze physical features of signals (i. e., signal powers, waveforms and frequency) [1, 46], and some need multiple receivers [24, 37] or devices to collaborate with each other [10]. As a consequence, these methods increase the cost of deploying anti-spoofing due to expensive equipment, computational complexity, and additional devices.

With these aforementioned concerns in mind, it becomes necessary to investigate new approaches for detecting GPS spoofing attacks specifically for the UAV context. Recently, with the remarkable progress of Computer Vision and Artificial Intelligence, some researchers have considered using vision-based techniques as assistance to detect GPS spoofing attacks. Such out-of-band methods have already been proved effective and successful [10, 33, 47, 68].

Liu et al. presented a UAV positioning method using priori remote-sensing information and SIFT-based (Scale Invariant Feature Transform) visual features [33]. The SIFT-based visual features of historical remote-sensing images were generated and stored on UAVs, so that when the UAVs are in the air, visual features of the environment can be compared with the historical remote-sensing image features to rectify the positioning. The authors suggested that CNN should be further studied, which is one of the contributions in DeepSIM, where our models are CNN deep learning neural networks for GPS spoofing detection. CNN-based models entitle

Table 8: Comparison of countermeasures against GPS spoofing attacks regarding deployment considerations. Difficulty refers to the overall evaluation of deploying corresponding countermeasures, and cost includes monetary, manpower and time cost.

Countermeasure	Difficulty	Cost	Redundant receivers	Specific HW	Compatibility with existing infrastructure	Automation
cryptographic schemes [13, 29, 41, 66]	High	High	No	No	Low	Yes
detection at signal level [1, 25, 35, 46, 48, 52, 57]	Medium	High	Partial	Yes	Medium	Yes
detection at direction of arrival sensing [24, 37, 59, 60]	High	Medium	Yes	No	Medium	Yes
existing out-of-band techniques, e. g., IMU [10, 33, 42, 47]	Medium	Medium	No	Yes	High	Yes
human-assisted method [68]	Medium	High	No	No	High	No
DeepSIM [*this paper*]	Medium	Low	No	No	High	Yes

DeepSIM to the better feature extraction and enable the system to evolve with the growth of data. Moreover, for DeepSIM, all computation tasks can be finished on the controller side, thus significantly improving the UAVs' battery life.

He et al. proposed a method to detect GPS spoofing based on the monocular camera and IMU sensor of a UAV through information fusion[10]. Essentially, the method is an improvement of IMU based GPS spoofing-detection methods, which leveraged velocity measured by IMU to verify the GNSS data. Specifically, the Lucas-Kanade (LK) method [34] was employed to estimate the velocity of UAV and then to compare with the velocity measured by IMU so that cumulative error in IMU measurement can be reduced. This method also requires on-board computation comparing with DeepSIM; however it could not eliminate systematically cumulative error due to its intrinsic shortcomings. Another similar method can be found in [47]. DeepSIM's simpler approach is to only use a camera in contrast to using both a camera and an IMU.

Zhu et al. proposed a collaborative approach for a single operator of multiple-UAV supervisory control systems using human-autonomy, in which human geolocation is used to help detect possible UAV cyber-attacks [68]. This method compares the non-tempered UAV camera video feed to the potentially falsified GPS location, and detect inconsistencies between the UAV video feed and UAV GPS location. Results indicate that only 65% of their experiments reach more than 80% accuracy of spoofing detection. Moreover, the success rate of prediction is not as high as ours (approximately 95%), their scheme is not so effective as DeepSIM either. According to their discussion, high task load results in more UAV damage due to higher mental workload. In addition, individual differences (e. g., video game experience or specific training) largely affect the success rate in detecting spoofing attacks. By contrast, we used the state-of-the-art techniques of deep learning in our approach. We trained machines rather than humans to identify GPS spoofing attacks. Our scheme can deal with hundreds of detection missions simultaneously in a short time. Last but not least, Zhu *et al.* did their experiment on a simulation platform. However, we conducted our experiment in a real scenario using real UAVs and all the data from the real world, which is more meaningful to reality.

The above proposals made innovative use of airborne equipment and resources of UAVs, e. g., cameras, maps, IMUs to design promising out-of-band ways to detect GPS spoofing attacks. For instance, the method proposed by Zhu et al. [68], however, relied on human detection and could not be fully automated. Others leveraged traditional hand-crafted visual features with performance impairment [33], or applied visual features to rectify data and compensate the drift from other sensors [10], requiring extra devices (i. e., IMUs) to cooperate with a camera. Table 8 provides a comparison of the deployment considerations of the countermeasures against GPS spoofing attacks.

8 CONCLUSION AND FUTURE WORK

In the near future, UAVs will play an increasingly significant role both in industrial and commercial realms. Unfortunately, UAVs are inherently prone to GPS spoofing attacks. In this paper, we presented *DeepSIM*, a deep-learning based approach for UAVs to detect GPS spoofing attacks. To the best of our knowledge, this paper is the first attempt to detect GPS spoofing attacks by determining a threshold of the similarity between satellite imagery and aerial photography with deep learning. In our work, we proposed four visual image anti-spoofing models. To train these deep learning models, we have constructed *SatUAV*, a pioneering model dataset. Experimental results show that our approach significantly outperforms previous human-assisted methods against GPS spoofing attacks. Furthermore, our method does not impose any modification on existing GPS infrastructure, structure of GPS signals, and does not need extra detection equipment.

For our future work, interesting research directions include: 1) studying the possibility of detecting GPS spoofing attacks in featureless or feature-poor areas, 2) optimizing parameters (parameter tuning) to get higher accuracy, 3) designing better models to overcome more complex challenges, e. g., detecting attacks at night, dealing with ephemeral objects, 4) carrying out more real experiments under different weather and air conditions (e. g., rain, snow, fog and haze) in more places for further evaluation, and 5) exploring to combine *DeepSIM* with time-series analysis or sensor fusion methods using not only cameras but also gyroscopes, IMUs, and accelerometers.

ACKNOWLEDGMENTS

The authors would like to thank Liangliang Cao, Qingtao Chen, Youtian Zhang, and Zhenjun Zhao for their helpful comments and the anonymous reviewers for their insightful suggestions. This work was supported by the Center for Cyber Security at New York University Abu Dhabi (NYUAD). This research was carried out on the High Performance Computing resources at NYUAD.

REFERENCES

[1] Dennis M Akos. 2012. Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC). *Navigation* 59, 4 (2012), 281–290.

[2] Daniele Borio and Ciro Gioia. 2016. A sum-of-squares approach to GNSS spoofing detection. *IEEE Trans. Aerospace Electron. Systems* 52, 4 (2016), 1756–1768.

[3] Andrea Cesetti, Emanuele Frontoni, Adriano Mancini, Primo Zingaretti, and Sauro Longhi. 2010. A vision-based guidance system for UAV navigation and safe landing using natural landmarks. *Journal of intelligent and robotic systems* 57, 1-4 (2010), 233.

[4] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cuDNN: Efficient Primitives for Deep Learning. *CoRR* abs/1410.0759 (2014). arXiv:1410.0759 http://arxiv.org/abs/1410.0759

[5] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 539–546.

[6] Samuel Dodge and Lina Karam. 2017. A study and comparison of human and deep learning recognition performance under visual distortions. In *26th International Conference on Computer Communications and Networks, ICCCN 2017.* Institute of Electrical and Electronics Engineers Inc.

[7] Alan Grant, Paul Williams, Nick Ward, and Sally Basker. 2009. GPS jamming and the impact on maritime navigation. *The Journal of Navigation* 62, 2 (2009), 173–187.

[8] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR '06).* IEEE Computer Society, Washington, DC, USA, 1735–1742. https://doi.org/10.1109/CVPR.2006.100

[9] Kim Hartmann and Christoph Steup. 2013. The vulnerability of UAVs to cyber attacks-An approach to the risk assessment. In *Cyber Conflict (CyCon), 2013 5th International Conference on.* IEEE, 1–23.

[10] Daojing He, Yinrong Qiao, Sammy Chan, and Nadra Guizani. 2018. Flight Security and Safety of Drones in Airborne Fog Computing Systems. *IEEE Communications Magazine* 56, 5 (2018), 66–71.

[11] Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence* 33, 12 (2010), 2341–2353.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[13] Guenter Hein, Felix Kneissl, Jose-Angel Avila-Rodriguez, and Stefan Wallner. 2007. Authenticating GNSS: proofs against spoofs, Part 2. *Inside GNSS* 2, 5 (2007), 58–63.

[14] Olli-Pekka Heinisuo. 2000. OpenCV: Open Source Computer Vision Library. https://opencv.org/.

[15] Hui Hu and Na Wei. 2009. A study of GPS jamming and anti-jamming. In *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference on*, Vol. 1. IEEE, 388–391.

[16] Todd Humphreys. 2012. Statement on the vulnerability of civil unmanned aerial vehicles and other systems to civil GPS spoofing. *University of Texas at Austin (July 18, 2012)* (2012).

[17] Todd E Humphreys, Brent M Ledvina, Mark L Psiaki, Brady W O'Hanlon, and Paul M Kintner Jr. 2008. Assessing the spoofing threat: Development of a portable GPS civilian spoofer. In *Proceedings of the ION GNSS international technical meeting of the satellite division*, Vol. 55. 56.

[18] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[19] Intel. 2020. *Intel Movidius Myriad 2 VPU Enables Advanced Computer Vision and Deep Learning Features in Ultra-Compact DJI Spark Drone.* Retrieved 2020-Oct-12 from https://www.movidius.com/news/intel-movidius-myriad-2-vpu-enables-advanced-computer-vision-and-deep-learn

[20] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 448–456.

[21] B Iyidir and Y Ozkazanc. 2004. Jamming of GPS receivers. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th.* IEEE, 747–750.

[22] Ali Jafarnia-Jahromi, Ali Broumandan, John Nielsen, and Gérard Lachapelle. 2012. GPS vulnerability to spoofing threats and a review of antispoofing techniques. *International Journal of Navigation and Observation* 2012 (2012).

[23] Kai Jansen and Christina Pöpper. 2017. Advancing Attacker Models of Satellite-based Localization Systems: The Case of Multi-device Attackers. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Boston, Massachusetts) *(WiSec '17).* ACM, New York, NY, USA, 156–159. https://doi.org/10.1145/3098243.3098270

[24] Kai Jansen, Matthias Schäfer, Daniel Moser, Christina Pöpper, and Jens Schmitt. 2018. Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks. In *IEEE Symposium on Security & Privacy.*

[25] Kai Jansen, Nils Ole Tippenhauer, and Christina Pöpper. 2016. Multi-receiver GPS spoofing detection: error models and realization. In *Proceedings of the 32nd Annual Conference on Computer Security Applications.* ACM, 237–250.

[26] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. 2020. imgaug. Retrieved 2019-Feb-01 from https://github.com/aleju/imgaug

[27] Alan Kim, Brandon Wampler, James Goppert, Inseok Hwang, and Hal Aldridge. 2012. Cyber Attack Vulnerabilities Analysis for Unmanned Aerial Vehicles. In *Infotech@Aerospace.*

[28] CG Leela Krishna and Robin R Murphy. 2017. A review on cybersecurity vulnerabilities for unmanned aerial vehicles. In *Safety, Security and Rescue Robotics (SSRR), 2017 IEEE International Symposium on.* IEEE, 194–199.

[29] Markus G Kuhn. 2004. An asymmetric security mechanism for navigation signals. In *International Workshop on Information Hiding.* Springer, 239–252.

[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. *ICLR Workshop* (2017). https://arxiv.org/abs/1607.02533

[31] Sven Lange, Niko Sunderhauf, and Peter Protzel. 2009. A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments. In *2009 International Conference on Advanced Robotics.* IEEE, 1–6.

[32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[33] XiJia Liu, XiaoMing Tao, YiPing Duan, and Ning Ge. 2018. Visual information assisted UAV positioning using priori remote-sensing information. *Multimedia Tools and Applications* 77, 11 (2018), 14461–14480.

[34] Bruce D. Lucas and Takeo Kanade. 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2* (Vancouver, BC, Canada) *(IJCAI'81).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 674–679.

[35] Mohsen Riahi Manesh, Jonathan Kenney, Wen Chen Hu, Vijaya Kumar Devabhaktuni, and Naima Kaabouch. 2019. Detection of GPS Spoofing Attacks on Unmanned Aerial Systems. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC).* IEEE, 1–6.

[36] Joel McNamara. 2008. *GPS for Dummies.* John Wiley & Sons. 59 pages.

[37] Paul Y Montgomery. 2011. Receiver-autonomous spoofing detection: Experimental results of a multi-antenna receiver defense against a portable civil GPS spoofer. In *Radionavigation Laboratory Conference Proceedings.*

[38] Thien Hoang Nguyen, Muqing Cao, Thien-Minh Nguyen, and Lihua Xie. 2018. Post-Mission Autonomous Return and Precision Landing of UAV. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV).* IEEE, 1747–1752.

[39] Tyler Nighswander, Brent Ledvina, Jonathan Diamond, Robert Brumley, and David Brumley. 2012. GPS software attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 450–461.

[40] Juhwan Noh, Yujin Kwon, Yunmok Son, Hocheol Shin, Dohyun Kim, Jaeyeong Choi, and Yongdae Kim. 2019. Tractor Beam: Safe-hijacking of Consumer Drones with Adaptive GPS Spoofing. *ACM Transactions on Privacy and Security (TOPS)* 22, 2 (2019), 1–26.

[41] Brady W O'Hanlon, Mark L Psiaki, Jahshan A Bhatti, Daniel P Shepard, and Todd E Humphreys. 2013. Real-Time GPS Spoofing Detection via Correlation of Encrypted Signals. *Navigation* 60, 4 (2013), 267–278.

[42] Gabriele Oligeri, Savio Sciancalepore, Omar Adel Ibrahim, and Roberto Di Pietro. 2019. Drive Me Not: GPS Spoofing Detection via Cellular Network. In *12th ACM Conference on Security and Privacy in Wireless and Mobile Networks.* Miami, FL, USA.

[43] Michael Ossmann and Great Scott Gadgets. 2016. Rapid Radio Reversing. *Black Hat Asia* (2016).

[44] Panagiotis Papadimitratos and Aleksandar Jovanovic. 2008. GNSS-based positioning: Attacks and countermeasures. In *Military Communications Conference, 2008. MILCOM 2008. IEEE.* IEEE, 1–7.

[45] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W.*

[46] Mark L Psiaki and Todd E Humphreys. 2016. GNSS spoofing and detection. *Proc. IEEE* 104, 6 (2016), 1258–1270.

[47] Yinrong Qiao, Yuxing Zhang, and Xiao Du. 2017. A Vision-Based GPS-Spoofing Detection Method for Small UAVs. In *2017 13th International Conference on Computational Intelligence and Security (CIS).* IEEE, 312–316.

[48] Aanjhan Ranganathan, Hildur Ólafsdóttir, and Srdjan Capkun. 2016. SPREE: A spoofing resistant GPS receiver. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 348–360.

[49] B Reavis and B Hem. 2011. Honeywell T-Hawk Aids Fukushima Daiichi Disaster Recovery: Unmanned Micro Air Vehicle Provides Video Feed to Remote Monitors. *Honeywell Aerospace Media Center* (2011).

[50] Logan Scott. 2001. Anti-spoofing & authenticated signal architectures for civil navigation systems. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*. 1543–1552.

[51] M Series. 2015. IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU* (2015), 2083–0.

[52] E Shafiee, MR Mosavi, and M Moazedi. 2018. Detection of Spoofing Attack using Machine Learning based on Multi-Layer Neural Network in Single-Frequency GPS Receivers. *The Journal of Navigation* 71, 1 (2018), 169–188.

[53] Scott Shane and David E Sanger. 2011. Drone Crash in Iran reveals secret US surveillance effort. *The New York Times* 7 (2011).

[54] Daniel P Shepard, Jahshan A Bhatti, Todd E Humphreys, and Aaron A Fansler. 2012. Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks. In *Radionavigation Laboratory Conference Proceedings*.

[55] S Alex Spelman. 2015. Drones: Updating the Fourth Amendment and the Technological Trespass Doctrine. *Nevada Law Journal* 16 (2015), 373.

[56] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[57] Minhong Sun, Yuan Qin, Jianrong Bao, and Xutao Yu. 2017. GPS Spoofing Detection Based on Decision Fusion with a K-out-of-N Rule. *IJ Network Security* 19, 5 (2017), 670–674.

[58] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. 1139–1147.

[59] Peter F Swaszek, Richard J Hartnett, Matthew V Kempe, and Gregory W Johnson. 2013. Analysis of a simple, multi-receiver GPS spoof detector. In *2013 International Technical Meeting of the Institute of Navigation*. 884–892.

[60] Peter F Swaszek, Richard J Hartnett, and Kelly C Seals. 2017. Using Range Information to Detect Spoofing in Platoons of Vehicles. In *Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017)*.

[61] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. 2011. On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 75–86.

[62] J Volpe. 2001. Vulnerability assessment of the transportation infrastructure relying on the global positioning system. In *Final Report*.

[63] Chris Wargo, Corey Snipes, Aloke Roy, and Robert Kerczewski. 2016. UAS industry growth: Forecasting impact on regional infrastructure, environment, and economy. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 1–5.

[64] Jon S Warner and Roger G Johnston. 2002. A simple demonstration that the global positioning system (GPS) is vulnerable to spoofing. *Journal of Security Administration* 25, 2 (2002), 19–27.

[65] Jon S Warner and Roger G Johnston. 2003. *Think GPS cargo tracking= high security*. Technical Report. Think Again. Technical report, Los Alamos National Laboratory.

[66] Kyle Wesson, Mark Rothlisberger, and Todd Humphreys. 2012. Practical cryptographic civil GPS signal authentication. *Navigation* 59, 3 (2012), 177–193.

[67] Yichi Zhang and Zhiyao Duan. 2017. IMINET: Convolutional semi-siamese networks for sound search by vocal imitation. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 304–308.

[68] Haibei Zhu, Mahmoud Elfar, Miroslav Pajic, Ziyao Wang, and ML Cummings. 2018. Human Augmentation of UAV Cyber-Attack Detection. In *Proceedings of the Human-Computer Interaction International Conference 2018*.

A APPENDIX – MODEL SELECTION

As mentioned before, we have explored four different deep learning models to detect GPS spoofing attacks. To accelerate the evaluation process, we experimented on a subset of our dataset. The subset consists of 599 image pairs. The experimental result can be found in Table 9.

A.1 Model 1: Distance Threshold

In Model 1, two images I_1 and I_2 are first sent to a pretrained ResNet to get their feature maps F_1 and F_2, respectively. Then, the Euclidean distance between the two feature maps is calculated. Finally, based on the comparison of this distance and the threshold, the model will determine whether the two images are from the same geographic location or not.

If the distance is larger than the threshold, it indicates that the two images are different, thus they are not from the same location. Otherwise, if the distance is smaller than the threshold, it indicates that the two images are similar and should come from the same location. The data was not split into training set and test set since there is no actual training. Every possible integer threshold was tested during our experiments, the results using the best threshold are reported in Table 9. Figure 9 shows the pipeline of this model.

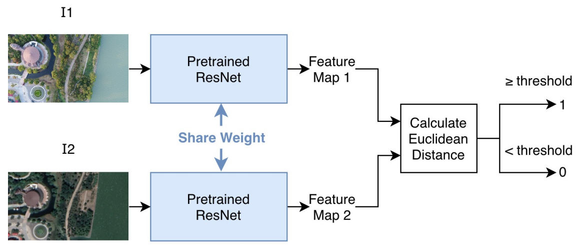


Figure 9: Network Structure of Model 1: Distance Threshold. The black arrows indicate data flow, there is no trainable component in this model.

```
Algorithm 3: Model 1: Distance Threshold
Input: Two images, $I_1$ and $I_2$
Output: 1 or 0, 0 indicates the images are from the same
        location, 1 indicates the opposite
Initialize: Pretrained $ResNet$ and threshold $T$;
$F_1 = ResNet(I_1)$;
$F_2 = ResNet(I_2)$;
$D = EuclideanDistance(F_1, F_2)$;
if $D \geq T$ then
    return 1;
end
return 0;
```

A.2 Model 3: Semi-Siamese Network

We adapted a Semi-Siamese Network [67] as the classification model. A pretrained ResNet was also leveraged to obtain the feature maps of I_1 and I_2. Thus, in place of directly applying threshold based linear discrimination, a 3-layer Fully Connected Network is attached to the end of the pretrained ResNet model to determine whether these two images are from the same geolocation. The loss function we use in Model 3 is Binary Cross Entropy loss, which is shown as follows:

$$\mathcal{L} = -(y \times log(p) + (1 - y)log(1 - p)), \tag{7}$$

Table 9: Experimental results of all candidates for the On-ground Model on a subset of the dataset.

Model	Dataset	Threshold	Accuracy	Precision	Recall	Error Rate	F1 Score
1: Distance Threshold	SatUAV-Suzhou	550	0.670	0.642	0.769	0.330	0.700
	SatUAV-Switzerland	423	0.781	0.836	0.700	0.219	0.762
2: Siamese ResNet	SatUAV-Training	2	0.931	0.914	0.935	0.069	0.924
	SatUAV-Test	2	**0.900**	0.905	0.931	**0.100**	**0.918**
3: Semi-Siamese Network	SatUAV-Training	0.5	0.912	0.856	0.992	0.088	0.919
	SatUAV-Test	0.5	0.892	0.859	0.953	0.108	0.904

where y denotes the label of two images, and p denotes the output of the network.

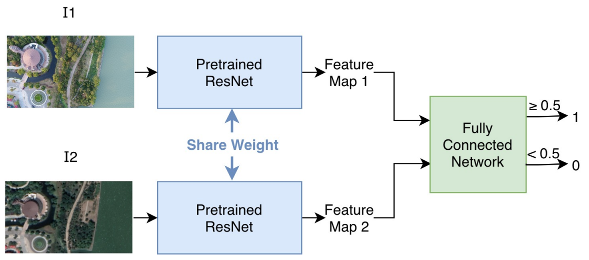


Figure 10: Network Structure of Model 3: Semi-Siamese Network. The black arrows indicate data flow the blue section is fixed and not trainable, the green section is a trainable network.

Figure 10 shows the pipeline of Model 3. As shown in the figure, the pretrained ResNet is fixed and will not be trained, only the Fully Connected Network will be trained during the whole training process. This technique is called *transfer learning* and saves a lot of computation time and memory usage for training.

A.3 Pros and Cons of Each Model

For Model 1, we have conducted experiments on the Suzhou and Switzerland subsets of the SatUAV. And for the trainable models, Model 2, 3 and 4, we divided the whole dataset into two parts, training data and test data. 80% of the dataset was randomly chosen to become training data, and the rest of them became test data. The hardware is the same as mentioned in Section 5.1. The experimental results of all candidates for On-ground Model are summarized in Table 9.

A.3.1 Model 1. Model 1 is a naïve model that does not require a high-end UAV controller to do the computation task, thus it is also faster than Model 2 and 3. Training is not needed, and the threshold of Model 1 is adjustable by the UAV operator.

The down sides of Model 1 outweigh its up sides. Model 1 is too naïve to perform well in most realistic scenarios. Its F1 score is much worse than Model 2 and 3. Model 1 is not trainable, which means it could not benefit much as data volume grows. In addition, there is one severe problem of Model 1 we observed as the experiments goes on. The best thresholds for different locations have large variations. As shown in Table 9, the best threshold value for Suzhou data is 550 while the best threshold value for Swiss data is 423. Hence, Model 1 lacks generalization, and is not as robust as Model 2 and 3.

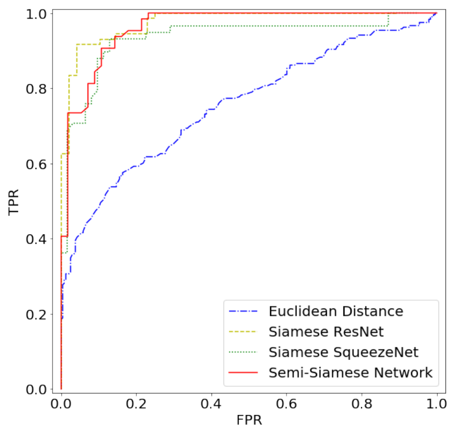


Figure 11: ROC curves of 4 models. Model 1 runs on SatUAV-Suzhou and SatUAV-Switzerland; Model 2, 3 and 4 run on SatUAV-Test.

A.3.2 Model 2 and Model 3. These two models are both trainable, which means they could evolve with more data collected for training. Unlike Model 1, Model 2 and Model 3 have much better generalization ability. One can train the model on SatUAV which only contains 6 cities but apply it globally on many human settlements. Similar to Model 1, there is an adjustable threshold in Model 2. But Model 3 has the best performance (F1 score) among all models. Though it is a little bit slower than Model 1, it is much faster in training than Model 2 and it does not require a high-end GPU server for training or inference. All training and testing of Model 3 can be done on a CPU server. Both models are slower, more time- and resource-consuming than Model 1. In addition, Model 2 requires a high-end GPU for efficient training.

A.3.3 Model 4. To balance the trade-off between accuracy and model complexity, we derived a new model from Model 2 that could run on the UAV's OBC independently without the assistance from the remote controller. Thus, we call it On-board Model. On-board Model is designed as a countermeasure of jamming attacks. If the attackers conduct a jamming attack to a UAV's remote control channel between the UAV and ground station, the UAV could still launch this On-board Model to detect spoofing attacks by itself. It is noteworthy that satellite imagery of scheduled flying area should be downloaded in advance to the UAV's storage before flying. However, the shortcoming is that accuracy and F1 score are reduced, compared with other complex models.

A.3.4 Summary. The Siamese ResNet and the Semi-Siamese Network get comparable performance. Both models outperform Model 1. The Siamese ResNet has the highest F1 score. However, the Semi-Siamese Network has much less trainable parameters, thus it can be trained in a short time even without a CUDA-enabled GPU. The biggest advantage of Model 4 is it can be run on-board. The Receiver Operating Characteristic curves (ROCs) of all the models are shown in Figure 11. Based upon the results, we chose Model 2—Siamese ResNet as the On-ground Model that runs on a ground station, and Model 4—Siamese SqueezeNet as the On-board Model that runs on a UAV.

To conclude, both Model 2 and 4 are good choices for DeepSIM. One can choose a model depending on the existing hardware resources and circumstances.