Security Analysis against Spoofing Attacks for Distributed UAVs

Kyo Kim*, Siddhartha Nalluri†, Ashish Kashinath*, Yu Wang†, Sibin Mohan*, Pajic Miroslav†, Bo Li*
*Department of Computer Science, †Department of Electrical and Computer Engineering
*University of Illinois at Urbana-Champaign, †Duke University
*{kkim103, ashishk3, sibin, lbo}@illinois.edu, †{siddhartha.nalluri, yu.wang094, miroslav.pajic}@duke.edu

***Abstract*—Distributed unmanned systems are increasingly finding use in a variety of applications, *viz.* reconnaissance, disaster management, search and rescue, *etc.* Sensing and actuation are key for the correct operation of such systems, especially since they do not have centralized control. These types of distributed systems have shown to be susceptible to spoofing attacks, such as false data injection attacks (on sensor values) — either via Man-in-the-middle (MitM) mechanisms or counterfeit signal generation. While machine learning techniques have been used to detect anomalous behavior, it has not found use in this domain. In this paper we pose the following questions: *(a)* "how well does a feed-forward deep learning model perform in detecting sensor anomalies?" and *(b)* "how can we inflate the dataset to reduce the cost of collecting data?" The second question aims to assist with the process of generating data for training the deep learning models and we study the effectiveness of Generative Adversarial Networks (GANs) for this purpose. Using software-in-the-loop (SITL) simulations, we analyze the feasibility of learning the behavior an unmanned autonomous vehicle (UAV). We present our findings for both of these questions in this paper.**

I. Introduction

Unmanned autonomous vehicles (UAVs) are canonical cyber-physical and IoT systems where a combination of physical and and computational components interact with the external environment in a feedback loop. However, if adversaries exploit the sensors in such systems then it is possible to move the system into an unsafe state or worse, compromise the integrity of the mission. This problem is exacerbated in the case of *distributed* UAVs that must often operate in a *decentralized* manner for the mission to succeed; for instance, applications such as search and rescue, reconnaissance, firefighting in remote areas, among others. Researchers have shown how to use out-of-band signal injection methods to compromise inertial sensors [19] and acoustic noise injection to attack both the gyroscope sensor [16] and the accelerometer sensor. Other attacks [7] demonstrate the ability to generate malicious network packets to compromise the integrity of the telemetry data. These attacks succeed due to the fact that sensor is data is either not authenticated or originates from an untrusted source.

There exists several techniques that incorporate authentication using cryptographic approaches [9], [10] and signal processing approaches [1], [15], primarily in the context of GPS. However, these approaches have limitations such as requiring customized hardware. They also lead to a large number of false alarms due to the lack of flexibility.

Researchers have attempted to add intelligence to the controllers (*i.e.,* the control code that estimates the current state of the system and decides on future actions) to detect or work around such attacks. Plugins have been proposed for the Extended Kalman Filter (EKF), the most common controller in contemporary UAVs (*e.g.,* [8], [13], [18]). However, EKF-based intrusion detection techniques have several shortcomings [18]: *(a)* they require precise knowledge of both the system model and the noise, which are not always possible in practice, and *(b)* they only work for linear systems, or nonlinear system working in a linearizable region. That is, if the system dynamics is highly nonlinear then the residual of the EKF can change even without attacks. Therefore, we argue that burdening the EKF with sensor spoofing detection is inherently a flawed approach. EKF is a state estimation and de-noising algorithm used to clean up sensor values and thereby aid in improving the stability of the system and was not designed to operate as an anomaly detector. As a result, we propose the *isolation of the security defense and the controller algorithm in distributed, decentralized UAV systems*.

In this paper, we propose the use of a *machine-learning technique to detect sensor spoofing* especially in UAVs. From a computational point of view, the advent of light-weight machine learning algorithms [14] has made it possible to run inference on resource-constrained systems that is characteristic of many UAVs. However, using deep learning approaches also raises issues with data collection. Since mission sensor values form a time series, collecting it is expensive (in terms of time required), both in simulation and in practice. Hence, we must augment the data to increase the size of the dataset for better training. Generative Adversarial Networks (GANs) [5] have shown promise in the image-domain in terms of generating realistic fake images. We explore the utility of GANs to make our dataset more complete. The process is to augment the data by generating synthetic datasets derived from the training set using GANs. However, we must first investigate if GAN performs "well enough" for generating such augmented data. We tested three GANs for this purpose: vanilla [5], softmax [11] and Wasserstein [2] and report their performance in Section V. Therefore, the goal of this paper is to investigate the practicality of using deep learning models in this domain for both, sensor anomaly detection and generating better data for training the models.

Workshop on Decentralized IoT Systems and Security (DISS) 2020
23 February 2020, San Diego, CA, USA
ISBN 1-891562-64-9
https://dx.doi.org/10.14722/diss.2020.23011
www.ndss-symposium.org

At a high level, our findings are: *(a)* the state of the mission (and especially changes between states) affects the performance of the anomaly detection model, *(b)* adversarial training aimed at reducing the effectiveness of adversarial inputs reduces the (average) prediction performance and *(c)* vanilla GAN generated samples that were recognizably close to the mission path but still not ideal. In summary, our contributions are twofold:

1) we evaluate the *robustness* of deep anomaly detection models for UAV swarms, especially under adversarial attacks and
2) use *domain-specific analyses* on the feasibility of using GANs to generate adversarial inputs as a method to test the overall robustness of the system.

As explained in the next section, we use a simple scenario with two UAVs to demonstrate our techniques and analyses.

II. Background

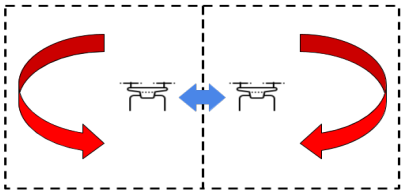


Fig. 1: Example of an intelligence, surveillance and reconnaissance (ISR) mission where each drone is circling in its designated area (red arrow). The blue arrow represents the auditing point where UAVs exchange sensor history.

A. Architecture of UAV and Ground Station

At a high level, the UAV consists of three components:
- *Low-level hardware* consisting of sensors, motors, propellers, batteries and so on,
- *Flight controller hardware* that includes NAVIO2 and Pixhawk controllers that serve a centralized interface to command and control low-level hardware,
- *Flight controller software* that provides high-level control to enable complex flight modes as well as other autonomy-related functionalities. (*i.e.,* PX4[1] and ArduPilot[2].

Both the low-level UAV hardware as well as the flight controller hardware can be simulated in software, *e.g.,* using *software-in-the-loop* (SITL) systems.

Mission. In a swarm of UAVs, we consider the case where each UAV is designated to survey a chosen area. This is typical of intelligence, surveillance, reconnaissance (ISR) missions. The UAVs then exchange information, including their history of sensor values, at fixed points in their trajectory. An example of such a mission (with two UAVs) is shown in Figure 1. If a UAV reports sensor readings that are not predicted by the model, then it is marked as anomalous by its neighbors. We chose this mission to demonstrate our methodology — our techniques and analyses can be applied to other missions involving distributed UAVs and distributed IoT systems in general.

Ground Station Architecture The ground station is a computer that runs ground control software that provides a user interface such as a map or coordinates to operate the UAV. Examples include Q Ground Control,[3] APM Planner 2.0[4] and Mission Planner.[5] In addition, the ground station often uses libraries such as dronekit[6] that enables the use of scripts to create and send messages to the UAV.[7] The UAVs communicate with each another and the ground station using the MAVLink (Micro Aerial Vehicle Link) Protocol,[8] a low-overhead messaging protocol — one that is commonly used in this domain. This protocol is specifically designed for resource-constrained devices such as UAVs as well as bandwidth-constrained links. The schematic architecture of this setup is shown in Figure 2.

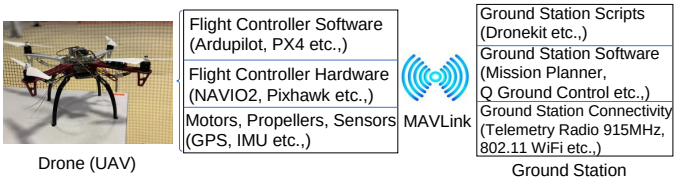


Fig. 2: Software and Hardware components present in a UAV and the Ground Station that communicate with one another using the MAVLink protocol.

B. Deep Models

Anomaly Detection Design Choices. Broadly, there are two basic approaches from the domain of machine learning: *(a)* model-based approach and *(b)* dataset-based approach. Model-based approach *engineers a model* that is designed to capture the dynamics of a particular UAV whereas data-driven approaches use a general model that *learns the dynamics* given a dataset. However, there exist trade-offs for using each approach [4]. The advantage for the latter is that there is no manual system identification, model validation and filter design. The drawback is that the human insight is not explicitly embedded into the model (*i.e.,* one must hope that the ML model learns the insight). However, by using a deep model, we aim to capture the nominal non-trivial dynamics of UAVs and achieve position prediction robustness under adversarial environment.

Stealthy Attacks. For deep models, adversarial examples are considered stealthy attacks since they are inputs that appear to be valid but contain small perturbations that cause the model to classify it differently. FGSM [6] is one of the methods that can generate adversarial examples. Within the context of this paper, the adversarial examples are inputs that increase the prediction error of the position given finite amount of perturbation from the valid input. We only know that during training and testing, certain examples cause error greater than the anomaly threshold. In an actual deployment, the error amount can be leveraged by the adversary to manipulate the drone; for instance, by making it drift off course while the drone still believes it to be on the right path. Therefore,

[1] https://px4.io/
[2] https://ardupilot.org/
[3] http://qgroundcontrol.com/
[4] https://ardupilot.org/planner2/
[5] https://ardupilot.org/planner/docs/mission-planner-overview.html
[6] https://dronekit.io/
[7] Note: while the main objective of the work is to analyze decentralized UAVs, we use a ground control station in our experiments for ease of capturing sensor data and carrying out analysis. The eventual goal of our work is to have all such analysis and detection running on each UAV.
[8] https://mavlink.io/en/

validating against the feasibility of *Stealthy Attacks* is crucial to the security analysis of such systems.

Generative Adversarial Networks (GANs) [5] have shown promise in generating realistic images in the computer vision domain. At a high-level, it works because two deep models compete against each other *viz.* a *generator* and a *discriminator*. The role of the generator is to fool the discriminator given a noise vector z. The role of the discriminator is to mark a given sample as fake (*i.e.,* is it from the generator) or real (*i.e.,* from the dataset). The loss incurred by the discriminator updates both, generator and the discriminator.

III. Our Approach

As mentioned earlier, we intend to use a feed-forward deep-learning model [17] to detect anomalous sensor signals, in particular those from the inertial measurement units (*e.g.,* accelerometer, gyroscope and GPS). We start with FGSM. We try other various methods to generate adversarial signals that can avoid detection by the deep-learning models in the future. Hence, we have two problems to address: *(a)* feasibility of the deep model and *(b)* use of GAN to generate adversarial data. Figure 3 shows the high-level pipeline of our approach. This section describes the specifics of each stage of the pipeline but we start with the adversary model.

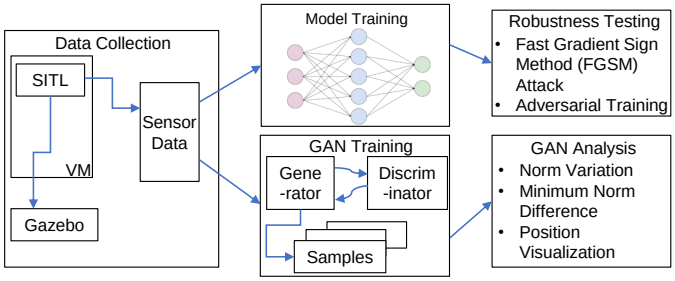


Fig. 3: We first collect sensor data directly from SITL that runs inside a VM to simulate a single core machine. We then use ML to capture the dynamics, develop a model for anomaly detection and train the GAN for data augmentation. We then analyze the feasibility of using both models.

Attack Model. In this paper, we consider false data injection attacks. We assume that the adversary has the capability to manipulate the sensor values (such as GPS, accelerometer and gyroscope) and these manipulated values are reflected in the audit by other drones. The attack vector we assume is similar to that of [3]: a malicious flight control library is on the system. However, we assume that the adversary can perform whitebox attacks on the model (*i.e.,* the adversary is aware of the parameters of the anomaly detector model). The adversary's objective is to manipulate the sensor values that cause the UAV to steer off of its original path and not raise any alarm by the model.

Data Collection. Sensor traces (from the IMU) were collected by hooking into the sensor generation code of the simulation platform (§IV). Mission traces, M, are composed of multiple trials T_i where i is the i^{th} trial. Each T_i is composed of sensor values x_j and the length of each T_i may not be the same. The feature length is 15 since each IMU provides 6 sensor values (each for the accelerometer and gyroscope) and the last three are the GPS values. Formally, the mission traces are in the following format: $M = \{T_0, ..., T_5\}, \quad T_i = \{x_0, .., x_{t_i}\} \in \mathbb{R}^{t_i \times 15}$

Deep Model. We consider multi-layer perceptron (MLP) with Leaky-Relu Activation [12] at each hidden layer for non-linearity. We use MLP as a baseline since it is the simplest deep model where the inference is more feasible in embedded and power-constrained devices compared more complex model such as convolutional or recurrent models. Leaky-Relu activation enables the model to learn non-linear dynamics. To test the effectiveness of the model, we test its performance using a clean dataset, *i.e.,* one with no adversarial signals. To test the robustness, we perform adversarial attack on the model using the Fast Gradient Sign method (FGSM). FGSM was the first method to generate an adversarial example and can be computed relatively fast compared to more complex methods. At high level, it is a method of perturbing the input such that the loss is maximized. Therefore, it serve as a baseline attack. We analyze the effectiveness of adversarial training on the model by performing the attack – *i.e.,* using the adversarially generated signals as input to the feed-forward model.

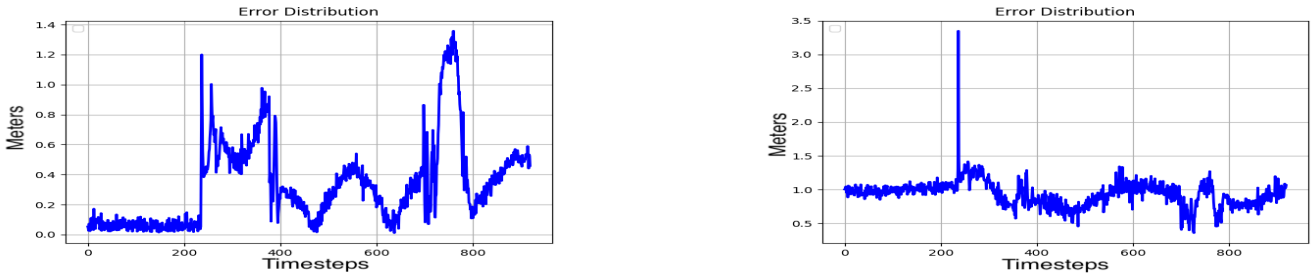
GAN. A generator, G, generates a segment of the sensor traces of length k with feature $d = 15$, $x' \in \mathbb{R}^{k \times d}$, given a random vector $z \sim \mathcal{N}[0, 1]^j$. The objective is for the G to capture the distribution of the mission – *i.e.,* the distribution of window of sensors that are expected in the mission. Generators are prone to learn only a portion of the total distribution (*i.e.,* mode collapse) which results in generating only that portion of the distribution. Also, training GANs may not be a stable process. Researchers have proposed the Wasserstien GAN [2] and Softmax GAN [11] variations to address these issues. Therefore, to evaluate the quality of G, we must first check if the generated examples are diverse (*i.e.,* the generated sensors are not too similar to each other) and similar to the true mission traces. Hence, we perform the following analysis (explained in section IV): *(a)* Norm variation analysis, *(b)* Minimum norm difference analysis and *(c)* Position visualization to measure the viability of quality of the generated samples quantitatively and qualitatively.

IV. Experimental Setup

To evaluate the performance our systems, we first describe the data collection setting and process and demonstrate that it is realistic. Then we provide details on how we built the anomaly detection model as well as the GAN. Finally, we describe the process of evaluating both.

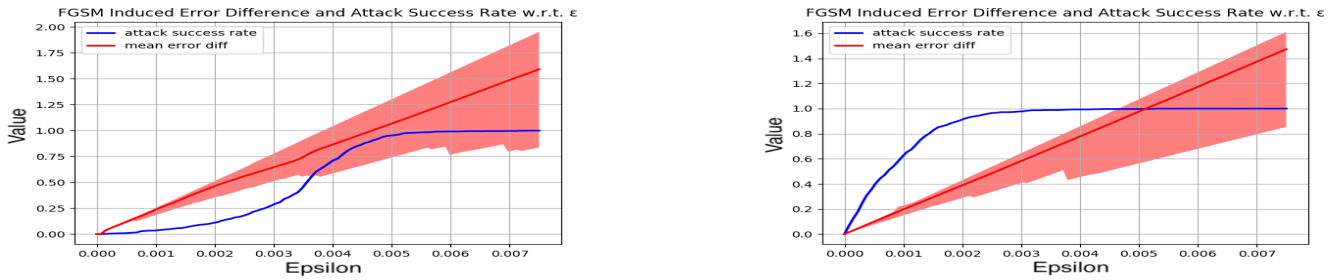
Simulation Setup consists of ArduPilot Software-in-the-Loop (SITL)[9] for flight control simulation and Gazebo[10] for physics simulation. We use the SITL since ArduPilot runs on the Navio2[11] platform and we can use the actual flight stack for our experiments while not having to deal with the vagaries of the actual UAV. This helps us refine our analysis and detection models before we implement it on a real UAV (we build the one in Figure 2 in our lab); our aim to demonstrate all of these techniques on a real system in the future. Since the default physics simulation of the SITL's is not sufficient, we

[9]https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html
[10]http://gazebosim.org/
[11]https://emlid.com/navio/



(a) (b)

Fig. 4: Distribution of error on testing set after adversarial training. X-axis is the mission duration in timesteps where each timestep is 200ms. Y axis is the error in meters. Before adversarial training (4a) and after adversarial training (4b)



(a) (b)

Fig. 5: X-axis is the epsilon amount of perturbation added to the input example. For blue line, Y-axis is the attack success rate. For the red line, Y-axis is the error difference in meters. the red shade is the range of error caused by the the FGSM. Figure 5a is the FGSM error induced and attack success rate before adversarial training. Figure 5b is after the adversarial traiing.

used the Gazebo interface, which is a well-established robotics simulator. The SITL runs inside a VM configured with a single core whereas Gazebo, physics simulator, runs on the host running the VM. Single core assignment to the VM was done to simulate the computational constraints of the UAV.

Mission Parameters To simulate an intelligence, surveillance and reconnaissance (ISR) mission, the UAV's mission is divided into: *(i)* 'liftoff' stage: a UAV elevates itself vertically by 10m, *(ii)* 'loiter' circle stage: UAV moves 10m outward and laps around the liftoff position with 10m radius and *(iii)* 'landing' stage: UAV moves inward and lands at a different position from where it started.

Data Preprocessing. To avoid numerical precision error, we converted the GPS values to the 3D Cartesian space where the origin is the UAV's initial position. The data is scaled using Scikit-learn[12] `MinMaxScaler` where each feature ranges between 0 and 1 as it is the standard deep learning practice.

Model Training. We consider a feed-forward neural net (*i.e.,* MLP) with 2 hidden layers with Leaky-Relu (LRelu) activation to make the model non-linear to enable learning the non-linear dynamics. The model attempts to predict the next position of the UAV and, hence, *learn the sensor dynamics of the mission*. Therefore, given the an example x_i at time step i, the label y_i is the position of x_{i+1}.

Model Evaluation. To evaluate how well the model predicts the next position, we use "prediction error distribution"". If the *model's prediction error is greater than 1 meter, we consider the sensor value to be anomalous*. To evaluate robustness, we will attack the model using FGSM.

[12]https://scikit-learn.org/stable/

GAN Training. We used the PyTorch implementation of three types (vanilla, Softmax and Wasserstein) of GANs[13] to train the generator. The generator is trained to produce an example of window size k. Therefore, the set of "real" examples for the discriminator is the sliding window size of k on the mission trial traces with stride 1. Formally, $X_{real} = \{a : a = T_i[j : j + k]\} \forall i < 5, j < t_i - k$.

GAN Evaluation. *Norm Variation Analysis* is an analysis on the variation in set of generated data. Formally, given a set of generated traces, $X' = x' : x' = G(z)$, we randomly select one of the generated data x'_c and compare the 2-norm difference against all other generated traces $x'_{c'} \in X'_{other} = X' \setminus \{x'_c\}$. We measure $std(||x'_c - x'_{c'}||) \forall x'_{c'}$

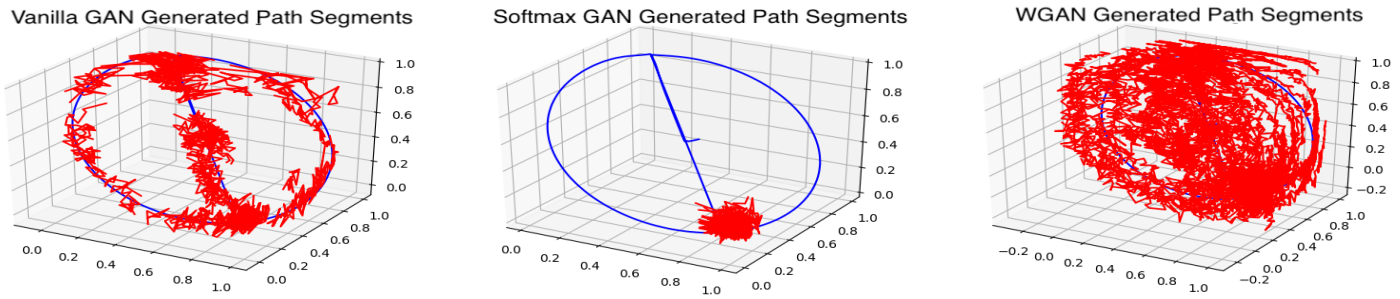
Minimum Norm Difference Analysis tells us how well G captured the distribution of a mission by evaluating the ideal location to splice the generated data into the original trial trace. Therefore, given that a mission, T, has length t and window of sensors, x, at timestep $i, \forall i \in [0, t - k]$, the min norm diff is defined as: $min\{||x' - x|| : x' \in X'\}$

Position visualization complements the previous two analyses since it is very difficult for humans to infer the state and stage of the UAV from raw sensor traces. Therefore, we 3D plot the generated samples' positions and compare it to the mission path.

V. Model and GAN Performance Evaluation

We now describe our results for evaluating the performance of our detection and adversarial data generation systems described above. The specific training parameters that we elaborate here are the best ones chosen after multiple trials.

[13]https://github.com/eriklindernoren/PyTorch-GAN



(a) (b) (c)

Fig. 6: 3D visualization of the path taken by the UAV (blue line) and path generated from vanilla GAN(6a), softmax GAN(6b), and WGAN(6c) (red line).

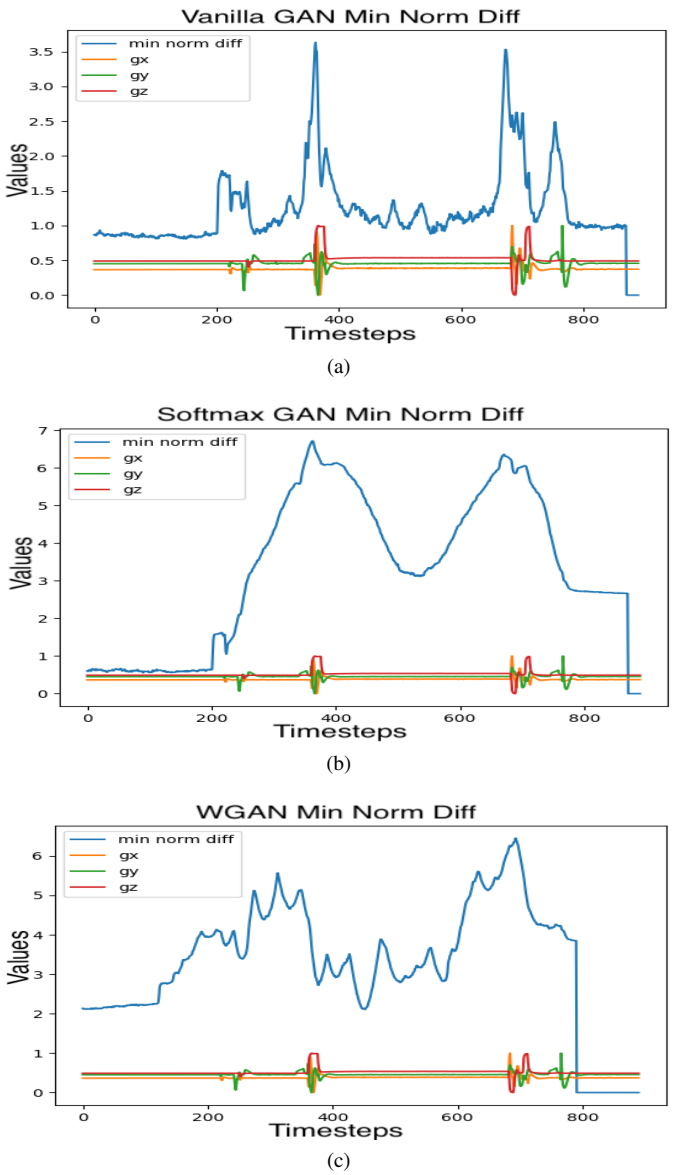
Model Performance: The MLP model was trained for 4800 epochs to achieve an average error of $0.332m$ on the test set with an accuracy 0.957 as shown in Figure 4a. The standard deviation for the error was 0.283 with median at 0.281. The figure indicates that a change in the mission stage causes a spike in the prediction error. For instance, at around $t = 235$, we see that the error jumps to 1.2m in figure 4a since that is when the UAV switches from initial mode to takeoff. Similarly, at around $t = 700$, the UAV switches from mode Loiter Circle to Landing. A false alarm is raised (*i.e.,* exceeds error threshold of 1m) when the UAV switches to lift off and landing modes as seen in the figure at around $t = 235$ and $t = 720$.

Attack on the Model. Figure 5a shows the FGSM attack rate before adversarial training. An attack is successful if the perturbation causes the input to be classified as anomalous but was previously not so. Instances that are near the error threshold are likely the ones that will cross over. For instance, when Epsilon is low, the timesteps with error rate close to the threshold (*e.g.,* $t = 700$) will likely to be the ones that will exceed the threshold. As shown in the figure, epsilon needs to be around 0.003 for the attack to be successful a quarter of the time.

After Adversarial training. The model's performance degrades to average meters error of 0.932 and 0.646 accuracy (threshold set to 1m error) with error distributed as shown in figure 4b. The average performance worsened likely due to the model needing to map wider range of inputs close to the labeled value. The error standard deviation was 0.190 with median at 0.947. Compared to Figure 4a, we see that the overall error values are higher while error variance is lower. Therefore, under the same threshold, we see that in Figure 5b the model is "more susceptible to the attack" since the post-adversarially trained model's error values are already near the threshold most of the time.

GAN Performance: The GANs (vanilla, Wasserstein and softmax) were trained with window size $k = 20$ and trained for 200 epochs. *Norm Variation Analysis.* Table I shows the statistics of the norm difference of 100 generated samples for each GAN. The low standard deviation of softmax indicates that it most likely suffers from mode collapse (*i.e.,* generated samples are similar) WGAN seems to generate more diverse examples (when compared to GAN) based on larger range in the "min" and "max" norm difference and higher mean and median values while maintaining similar standard deviation.

Minimum Norm Difference. Figures 7 show the relationship



(a) (b) (c)

Fig. 7: Min norm difference vs gyro sensor reading comparison: (7a) GAN, (7b) Softmax GAN, and (7c) Wasserstein GAN. The x-axis is the mission duration where 1 timestep is 200ms. The y-axis for the blue line is the 2-norm difference value and the sensor values. In all three GANs, we see that changes in gyro reading results in spike in norm difference.

GAN Type	Vanilla	Softmax	Wasserstein
Max	3.411562	1.593482	8.119476
Min	0.149875	0.447159	0.412769
Mean	0.149875	0.924011	4.596173
Median	3.672165	0.906382	5.025689
Std	2.192810	0.271916	2.060317

TABLE I: Norm Difference Statistics

between the minimum norm difference and the gyroscope sensor values. The spikes indicate that all three GANs have a difficult time capturing the segment of sensor values where big changes are occurring in the IMU readings. The sudden spikes indicate changes in mission phases: *(a)* the first spike is the liftoff, *(b)* second is the start of the circle loitering phase, *(c)* the third is the end of the loiter while *(d)* the last spike is the landing stage. This behavior seems to indicate that capturing the sensor distribution when the mode changes occur is hard and *may require separate models for each stage of the mission*.

Position Visualization. Figure 6 plots a visualization of the position of the UAV — based on the generated signals from each of the GAN models. Each axes are the basis in the 3D space. The figure (6b) highlights the mode collapse in Softmax GAN as analyzed using norm variation analysis. The visualization also confirms that WGAN generated much more diverse examples (*i.e.,* wider range of generated samples) However, the WGAN does not seem to capture the sensor distribution of the mission as it looks like it is generating random positions. Of the three, vanilla GAN seems to capture the distribution of the mission the best because the generated path aligns closest to the mission path. However, it does not seem to provide adequate performance (*i.e.,* generating a window of realistic mission trace) to be used in data augmentation.

Discussion Hence the main takeaways from this work include:
1. a change in the UAV mission state affects the performance of the model; hence we might need separate models for each phase.
2. adversarial training results in a degradation of the average prediction performance; however, the prediction performance becomes more consistent.
3. vanilla GAN performed best (of the three we tested) and was able to generate samples that were recognized as being close to the mission path but not ideal.

VI. Conclusion

We demonstrated the practicality of deep models for UAV sensor anomaly detection. We developed a feed-forward deep model that captures the dynamics of the UAV and GAN to augment the dataset for better training. Our early results show that recreating the models for even simple missions (especially during transition stages) is hard but vanilla GANs worked best for this task. In addition, adversarial training provides stable prediction error at the expense of performance.

Acknowledgment

This work was supported by Boeing Research & Technology (BR&T) under the Collaborative Research Project BRT-Z0418-5048: Machine Learning-based Communication and Anomaly Detection in Distributed Autonomous UAV Swarms.The authors would like to thank Dr. Jae H. Kim (Boeing PM) for his advice and guidance throughout the project. The authors would also like to thank AMD for providing the Vega 20 GPU which was used in our experiments.

References

[1] D. M. Akos, "Who's afraid of the spoofer? gps/gnss spoofing detection via automatic gain control (agc)," *NAVIGATION: Journal of the Institute of Navigation*, vol. 59, no. 4, pp. 281–290, 2012.

[2] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[3] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Out of control: stealthy attacks against robotic vehicles protected by control-based techniques," in *ACSAC*, 2019.

[4] P. Freeman, R. Pandita, N. Srivastava, and G. J. Balas, "Model-based and data-driven fault detection performance for a small uav," *IEEE/ASME Transactions on mechatronics*, vol. 18, no. 4, pp. 1300–1309, 2013.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] K. Highnam, K. Angstadt, K. Leach, W. Weimer, A. Paulos, and P. Hurley, "An uncrewed aerial vehicle attack scenario and trustworthy repair architecture," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE, 2016, pp. 222–225.

[8] Y. Huang, J. Tang, Y. Cheng, H. Li, K. A. Campbell, and Z. Han, "Real-time detection of false data injection in smart grid networks: An adaptive cusum method and analysis," *IEEE Systems Journal*, 2014.

[9] T. E. Humphreys, "Detection strategy for cryptographic gnss anti-spoofing," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 1073–1090, 2013.

[10] M. G. Kuhn, "An asymmetric security mechanism for navigation signals," in *International Workshop on Information Hiding*. Springer, 2004, pp. 239–252.

[11] M. Lin, "Softmax gan," *arXiv preprint arXiv:1704.06191*, 2017.

[12] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[13] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE transactions on control of network systems*, 2014.

[14] J. Manning, D. Langerman, B. Ramesh, E. Gretok, C. Wilson, A. George, J. MacKinnon, and G. Crum, "Machine-learning space applications on smallsat platforms with tensorflow," in *Proceedings of the 32nd Annual AIAA/USU Conference on Small Satellites, Logan, UT, USA*, 2018, pp. 4–9.

[15] A. Ranganathan, H. Ólafsdóttir, and S. Capkun, "Spree: A spoofing resistant gps receiver," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 348–360.

[16] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015.

[17] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[19] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.