ADS-B Plane Spoofing Detection using Neural Networks and RF Fingerprinting

Angelina Tsuboi

Mathematics & Computer Science

03/02/2023

Dr. Dijanna Figueroa

Chadwick School

Los Angeles, California

Abstract

In the aviation industry, aircraft spoofing is a growing concern as the use of Automatic Dependent Surveillance-Broadcast (ADS-B) technology becomes increasingly widespread. To combat this threat, I developed a device and a web app that detects spoofed aircraft using a convolutional neural network trained on a dataset of verified and spoofed ADS-B signals. The implementation is two-fold: a low-cost device made with the Raspberry Pi, a FlightAware RTL-SDR, and a 1090MHz antenna-based tailored for research that uses a Convolutional Neural Network and RF Fingerprinting techniques to identify and flag suspicious signals, and a Flask-based web application that uses the same algorithm to detect spoofed aircraft from any location using an API, and displays the information onto an intuitive map layout. After testing the implementations on multiple flights in a single-engine aircraft throughout Los Angeles and possible attack scenarios in a simulated environment, both the device and the web app have a real-time detection system with an accuracy rate of over 90% and protect against potential flight delays and mid-air collisions caused by fake aircraft signals.

Table of Contents

Introduction

Air traffic control is critical for ensuring safe and efficient air travel, but it's not without its flaws. One such flaw is the use of Automatic Dependent Surveillance-Broadcast (ADS-B) technology, which is vulnerable to ground-based cyberattacks that could result in spoofed or fake aircraft being broadcasted in the sky (Amin, Clark, Offutt and Serenko, 2014). The consequences of such attacks could be devastating, including massive flight delays and even mid-air collisions. As a pilot concerned with the safety of air travel, I decided to take action by developing a low-cost device called Fly Catcher that detects spoofed aircraft in real-time.

ADS-B is used by pilots and air traffic control to track aircraft as they fly by broadcasting an aircraft's position, speed, altitude, and other information in real-time. This information is received by air traffic controllers and nearby aircraft, allowing them to "see" each other and avoid collisions. However, ADS-B messages are unauthenticated and unencrypted, making it vulnerable to ground-based cyberattacks that could result in fake or spoofed aircraft being broadcasted in the sky (Wang, Zou, and Ding, 2020). To address this growing concern, I developed a hardware prototype called Fly Catcher that detects spoofed aircraft by analyzing the ADS-B data received by the device and by using a technique called RF Fingerprinting (Jian, Tong, and Rendon, 2020). The ADS-B data is passed through a decoder and then into a Convolutional Neural Network written with Python and trained on a dataset of valid ADS-B signals as well as a generated

spoofed set of aircraft signals to detect whether the aircraft is spoofed or not (Ying, Mazer, and Bernieri, 2019). The result outputted by the neural network is then displayed onto a radar screen on the Raspberry Pi, allowing users to detect spoofed aircraft near them.

Alongside the device, I also created a proof of concept web application retrieves live ADS-B data from live data sources like Open Sky to provide a live coverage of aircraft integrity at any specified location by inputting the user's location into a map graphical interface and passing in the ADS-B data within a 100 nautical mile radius of that location into the Neural Network (Meides, 2023).

Throughout my research, I aimed to answer the following question: "How can a Convolutional Neural Network in conjunction with RF Fingerprinting be trained to detect spoofed aircraft by analyzing the ADS-B properties of an aircraft with a high degree of accuracy?" My objective is to demonstrate the effectiveness of Fly Catcher in detecting spoofed aircraft by conducting a series of tests both on the ground and in flight. My hypothesis is that the Neural Network algorithm will be able to detect spoofed aircraft with an accuracy rate of above 90% by implementing a rigorous neural network model that analyzes aircraft telemetry and an RF Fingerprinting method that examines the RSSI received by each aircraft signal (Haoran, Zha, and Tian, 2020).

Materials

This project has two implementations: the web app and the physical device. You can use either option to set up the project, however, the web app implementation is easier to set up and does not require any tangible materials.

Device Materials

- 1090MHz Rubber Ducky Antenna
- Raspberry Pi 3B
- FlightAware Pro Stick Plus SDR
- 3.5 in TFT Screen
- Portable Battery Charger
- USB-C to Micro USB Cable
- Custom 3D Printed Case (optional)
- SD Card
- Rasbian Operating System
- 4x 3/32 Screws
- Python and Pip on Raspberry Pi
- TensorFlow, Keras, and Pandas Libraries

Web App Materials

- Computer
- Python, Pip, and Flask
- TensorFlow, Keras, and Pandas Libraries

Software Used

- OpenSky (Meides, 2023)
- FlightAware (Flight tracker / flight status, 2023)
- ADSB Exchange (Meides, 2023)

Libraries Used

- Numpy
- Keras
- SkLearn
- Pandas
- Tensorflow
- Matplotlib
- CSV
- JSON

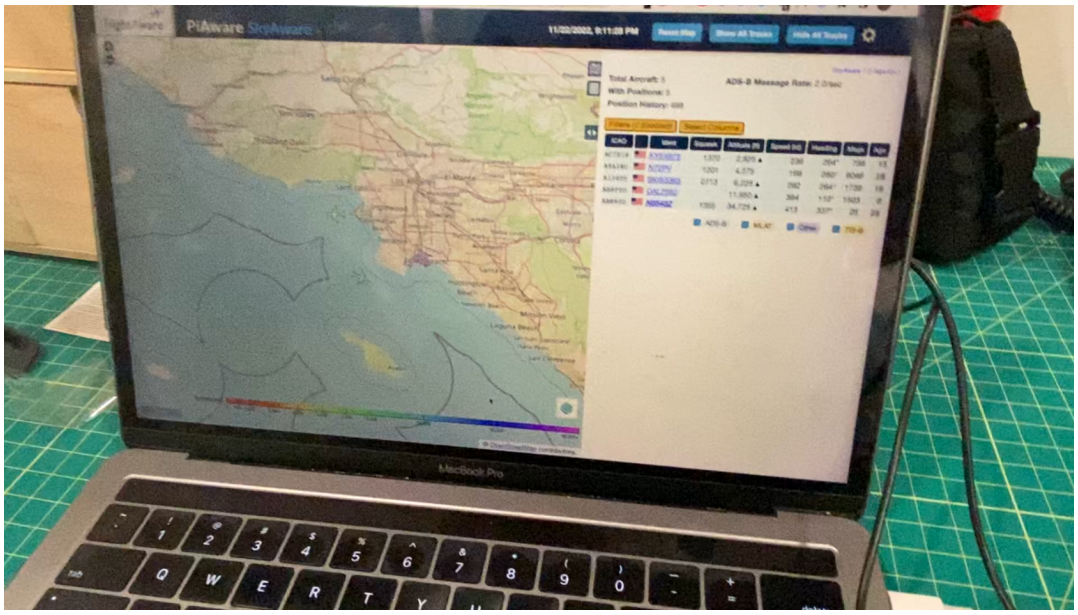
Procedure

The sequence of creating the ADS-B Spoof Detection device and conducting research was split into three sequential steps: device creation, data collection, neural network development, and web app creation. I began the project by developing a proof of concept hardware device to analyze ADS-B information using the Raspberry Pi. To create the working prototype of the device I followed the procedure outlined below.

Device Creation

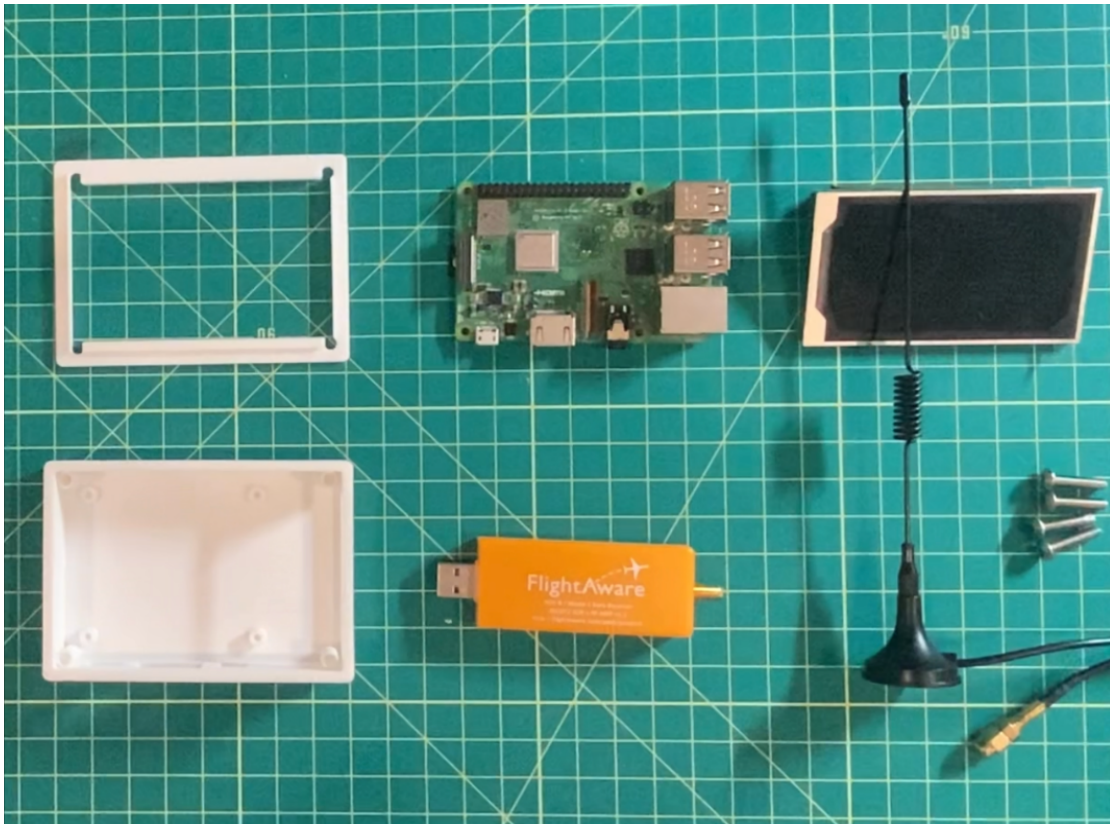
1. Gather all the necessary materials to construct the device
2. Install the Rasbian operating system to the Raspberry Pi with the SD Card
3. Connect the Flight Aware SDR to the Raspberry Pi using the Micro USB cable
4. Connect the 1090 MHz antenna to the Flight Aware SDR
5. Configure the 3.5-inch TFT Screen to the Raspberry Pi
6. Place the Device into the 3D Printed Case
7. Ensure Python and Pip are installed on the Raspberry Pi
8. Install dump-1090 FlightAware library on the Raspberry Pi to receive ADS-B information
9. Create a Python program that aggregates and tabulates ADS-B data from the receiver to a CSV File
10. Download the Neural Network code into the device

11. Feed the aircraft data retrieved by the receiver into the Neural Network
12. Create a Graphical Display using PyGame to display nearby aircraft and indicate whether they are spoofed or not



Localhost server hosted by the device showing aircraft telemetry

1090MHz antenna of the device collecting data from my room



Materials used to construct the device

Picture of the completed build

Display shown on the TFT Screen

Data Collection

1. Configure FlightAware localhost instance on device to output ADS-B data
2. Parse ADS-B data from the receiver by sending GET HTTP requests
3. Create a Python Program that converts JSON ADS-B data into a CSV file
4. Save the CSV data into the SD Card on the device
5. Collect data from home by leaving the device on for an extended period of time
6. To increase the range of the dataset, I brought the device with me on a flight across Los Angeles

7. I brought the device with me on multiple trips to different destinations such as Santa Monica, Malibu, Torrance, DTLA, Burbank, etc. to collect data

Device in back of airplane collecting data

Device shown with the SportCruiser I flew to collect data

Setting up aircraft to house the device

Neural Network Development

1. Gathered additional data from sources such as ADSB Exchange, FlightAware, and Open Sky
2. Generate spoofed aircraft data using a custom Python program
3. Stored all the ADS-B data into a single, unified CSV file
4. Processed the data using Numpy and reshaped it to match the input format
5. Split the data into training and testing sets
6. Created a CNN architecture using Keras adjusting layers, weights, and techniques to optimize on accuracy
7. Displayed the accuracy and other statistics of model results using Matplotlib

In [10]:
```python
# CNN Architecture
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_sh
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', met

# Train the model on the training data
model.fit(x_train, y_train, epochs=10, batch_size=32)

# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)

# Use the model to classify new ADSB data
```

Initial CNN architecture code with Python

```python

DUMP1090DATAURL = "http://localhost:8080/data/aircraft.json"

class FlightData():
    def __init__(self, data_url = DUMP1090DATAURL):

        self.data_url = data_url

        self.refresh()

    def refresh(self):
        #open the data url
        print("http://localhost:8080/data/aircraft.json")

        self.req = urlopen("http://localhost:8080/data/aircraft.json")

        #read data from the url
        self.raw_data = self.req.read()
        print(self.raw_data)

        #encode the data
        encoding = self.req.headers.get_content_charset()

        #load in the json
        self.json_data = json.loads(self.raw_data.decode('utf-8'))

```

Python program to parse ADS-B data

Web App Creation

1. Initialize a Flask server
2. Configure the OpenSky API to retrieve live ADS-B information
3. Display the ADS-B information on a map display
4. Feed the ADS-B information into the CNN on the program to detect whether the aircraft is spoofed or not
5. Indicate whether the aircraft is spoofed on the map layout

```
94      def dot_add(self, dot_id, x, y, data = None, back_col = None, dot_col = None):
95          if back_col == None: back_col = self.back_col
96          if dot_col == None: dot_col = self.radar_col
97          #work out x, y on screen
98          screen_x, screen_y = self._calc_screen_x_y(x, y)
99
100         #does the dot already exist?
101         if dot_id in self.dots:
102             dot = self.dots[dot_id]
103             dot.move(screen_x, screen_y)
104         else:
105             dot = RadarDot(self.screen, screen_x, screen_y, 10, data = data, back_col = back_col, dot_col = d
106             self.dots[dot_id] = dot
107
108     def dot_remove(self, dot_id):
109         if dot_id in self.dots:
110             del self.dots[dot_id]
111
112     def dot_move(self, dot_id, x, y):
113         screen_x, screen_y = self._calc_screen_x_y(x, y)
114         self.dots[dot_id].move(screen_x, screen_y)
115
116     def dot_move_by(self, dot_id, x, y):
117         self.dots[dot_id].move_by(x, y)
118
119     def _calc_screen_x_y(self, x, y):
120         diff_x = (x - self.radar_pos[0]) * self.scale
121         diff_y = (y - self.radar_pos[1]) * self.scale
122         screen_x = int(round(self.centre[0] + diff_x, 0))
123         screen_y = int(round(self.centre[1] + diff_y, 0))
124         return screen_x, screen_y
125
126     def _display_dots_on_line(self, x1, y1, x2, y2):
127         for key, dot in self.dots.copy().items():
128             if dot.fade_step == 0 or dot.fade_step > 50:
129                 intersect, pos = dot.on_line(x1, y1, x2, y2)
130                 if intersect == True:
131                     dot.show()
132
133     def dot_at_point(self, point):
134         dot_found = None
135         for key, dot in self.dots.copy().items():
136             if dot.rect.collidepoint(point):
137                 dot_found = key
138         return dot_found
```

Code snippet of the web app

Data Collection

In order to collect data for research, RF Fingerprinting analysis, and to train the Convolutional Neural Network, I underwent three stages of data collection: ADS-B data retrieval from a home-based station, data aggregation from online sources, and data collection from aircraft flights throughout Los Angeles.

Home-based ADS-B Station

In order to collect data from a home station, I set up the device in my room and recorded the ADS-B information of nearby aircraft using the 1090 MHz antenna and storing that data into a CSV file. The total device operation time at my home-based ADS-B station was around 20 hours giving me a wide range of aircraft data and scenarios to analyze. I wrote a program to format the JSON ADS-B output into a CSV file.

ROW #	HEX	TYPE	FLIGHT	R	T	ALT_BARO	GS	TRACK	BARO_RATE	LAT	LON	NIC	RC	SEEN_POS
# 1	845f9f	adsc	KZ51	JA11KZ	B748	31996	487.0	244.0	48	57.32872	-177.562752	0	0	873.399
# 2	86d624	adsc	ZG23	JA825J	B788	41000	481.0	259.0	-16	56.712799	-176.982708	0	0	276.096
# 3	a835c5	adsc	UP34	N628UP	B748	34004	485.0	218.13	16	2.205334	-174.191151	0	0	1163.4
# 4	4d0104	adsc	CV4316	LX-VCD	B748	33996	505.0	67.42	0	57.335758	-173.682518	0	0	836.517
# 5	86e4da	adsc	JL11	JA868J	B789	38000	494.0	258.2	0	58.272343	-173.583469	0	0	384.799
# 6	71c083	adsb_icao	KAL074	HL8083	B789	38000	487.9	241.07		61.398605	-172.388142	8	186	30.917
# 7	76cdb9	adsc	SQ31	9V-SMY	A359	37000	507.0	259.25	0	31.667919	-172.002811	0	0	5.538
# 8	7c1479	adsc	QF56	VH-EBV	A332	36000	477.0	232.29	0	20.039577	-170.449963	0	0	615.477
# 9	86ebb6	adsb_icao	ANA135	JA886A	B789	35975	508.3	268.2		58.778811	-170.167419	8	186	5.589
# 10	abba6c	adsc	FX73	N855FD	B77L	34000	474.0	210.95	0	2.224903	-169.668388	0	0	199.17
# 11	7c531d	adsc	QF104	VH-QPB	A333	36000	453.0	214.54	-16	5.316525	-169.059849	0	0	16.53
# 12	ab271f	adsb_icao	AAL61	N818AL	B788	38000	496.9	250.36		61.346446	-168.022359	8	186	3.382

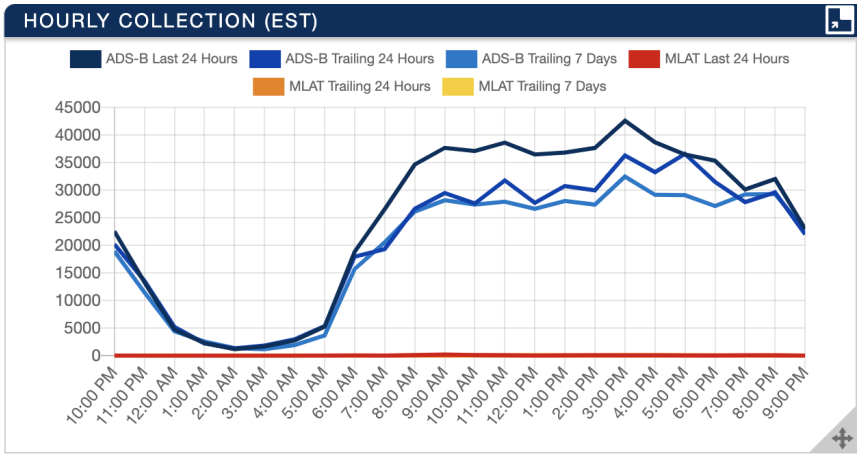
CSV format of data recorded at home-based station

{"aircraft": [{"hex": "ADCF46", "type": "adsc", "flight": "R658", "r": "YBMZA3", "t": "IL96", "alt_baro": 14981, "gs": 992, "track": 329, "baro_
-176.2820790187022, "nic": 0, "rc": 0, "seen_pos": 9, "mlat": [], "tisb": [], "messages": 905, "seen": 547, "rssi": -33, "is_spoofed": true}, {"
"ERJ145", "alt_baro": 28624, "gs": 857, "track": 58, "baro_rate": 1120.6695867555136, "lat": -70.63707032573711, "lon": 155.43889428300025, "nic
"seen": 407, "rssi": -121, "is_spoofed": true}, {"hex": "B96D38", "type": "adsc", "flight": "IMGI", "r": "F4VWJ7", "t": "TBM9", "alt_baro": 1848
54.975601877867916, "lon": 42.73213053117237, "nic": 0, "rc": 0, "seen_pos": 26, "mlat": [], "tisb": [], "messages": 191, "seen": 105, "rssi": -
"QNYO", "r": "DP0YEU", "t": "A340", "alt_baro": 31991, "gs": 230, "track": 256, "baro_rate": 1158.550622550522, "lat": 9.09813828203248, "lon":
"tisb": [], "messages": 175, "seen": 913, "rssi": -60, "is_spoofed": true}, {"hex": "B9FE57", "type": "adsc", "flight": "QLKJ", "r": "L0H7NV",
-1892.060306911024, "lat": -4.236108950601334, "lon": 82.53803360858547, "nic": 0, "rc": 0, "seen_pos": 7, "mlat": [], "tisb": [], "messages": 4
"type": "adsc", "flight": "DWP3", "r": "0TNKZ1", "t": "A320", "alt_baro": 16748, "gs": 482, "track": 217, "baro_rate": 1437.5175835079049, "lat"
"seen_pos": 2, "mlat": [], "tisb": [], "messages": 857, "seen": 67, "rssi": -59, "is_spoofed": true}, {"hex": "B0BB00", "type": "adsc", "flight"
"track": 296, "baro_rate": -127.71259099474446, "lat": -33.70444142432079, "lon": -133.265091928988, "nic": 0, "rc": 0, "seen_pos": 14, "mlat":
"is_spoofed": true}, {"hex": "D25EC6", "type": "adsc", "flight": "L5TF", "r": "9BAYFM", "t": "B767", "alt_baro": 3818, "gs": 583, "track": 202,
-174.51480503802466, "nic": 0, "rc": 0, "seen_pos": 16, "mlat": [], "tisb": [], "messages": 642, "seen": 260, "rssi": -138, "is_spoofed": true},
"t": "F35", "alt_baro": 1678, "gs": 766, "track": 57, "baro_rate": 181.78840537667838, "lat": 84.31849691882024, "lon": -127.98286993054914, "ni
493, "seen": 282, "rssi": -159, "is_spoofed": true}, {"hex": "8921E5", "type": "adsc", "flight": "7WFB", "r": "QA0PEF", "t": "G650", "alt_baro":
-19.618244142127736, "lon": -104.36195423053918, "nic": 0, "rc": 0, "seen_pos": 14, "mlat": [], "tisb": [], "messages": 348, "seen": 159, "rssi"
"flight": "PQQG", "r": "JZK6RU", "t": "G650", "alt_baro": 10008, "gs": 352, "track": 329, "baro_rate": 1740.4398243376854, "lat": -79.3492047098
"mlat": [], "tisb": [], "messages": 15, "seen": 944, "rssi": -157, "is_spoofed": true}, {"hex": "908E48", "type": "adsc", "flight": "7FCW", "r":
"baro_rate": -1187.4904254666803, "lat": 37.64339471744303, "lon": -142.1751194570225, "nic": 0, "rc": 0, "seen_pos": 18, "mlat": [], "tisb": []
{"hex": "C563F3", "type": "adsc", "flight": "NCKI", "r": "WVEL8D", "t": "ERJ145", "alt_baro": 27884, "gs": 882, "track": 103, "baro_rate": -358.
"nic": 0, "rc": 0, "seen_pos": 30, "mlat": [], "tisb": [], "messages": 624, "seen": 780, "rssi": -18, "is_spoofed": true}, {"hex": "6D610E", "ty
27694, "gs": 777, "track": 314, "baro_rate": -758.8941505361731, "lat": 69.4591185015334, "lon": -58.736257977924, "nic": 0, "rc": 0, "seen_pos"
-117, "is_spoofed": true}, {"hex": "C59AB2", "type": "adsc", "flight": "98JU", "r": "A396JE", "t": "UH60", "alt_baro": 13733, "gs": 531, "track"
"lon": 55.22126849034464, "nic": 0, "rc": 0, "seen_pos": 5, "mlat": [], "tisb": [], "messages": 415, "seen": 272, "rssi": -90, "is_spoofed": tru
"t": "Cessna Citation X", "alt_baro": 27179, "gs": 128, "track": 241, "baro_rate": -623.4248849558401, "lat": 89.72702163236468, "lon": 60.43773
"messages": 176, "seen": 759, "rssi": -138, "is_spoofed": true}, {"hex": "337D56", "type": "adsc", "flight": "2TI8", "r": "WJVGP3", "t": "B777",
-1769.8393416672604, "lat": 46.41044468542532, "lon": -35.11638185309354, "nic": 0, "rc": 0, "seen_pos": 4, "mlat": [], "tisb": [], "messages":
"type": "adsc", "flight": "RLHG", "r": "NQU4YB", "t": "A330", "alt_baro": 680, "gs": 926, "track": 300, "baro_rate": -1532.3074175103684, "lat":
"seen_pos": 6, "mlat": [], "tisb": [], "messages": 608, "seen": 234, "rssi": -169, "is_spoofed": true}, {"hex": "25E0C9", "type": "adsc", "fligh
"gs": 578, "track": 300, "baro_rate": -1509.2089648720805, "lat": 1.4849833049112675, "lon": 18.55896956914819, "nic": 0, "rc": 0, "seen_pos": 1
"is_spoofed": true}, {"hex": "AD1C9F", "type": "adsc", "flight": "H7V1", "r": "HFYBST", "t": "XT6", "alt_baro": 22891, "gs": 347, "track": 251,
-16.22979811697155, "nic": 0, "rc": 0, "seen_pos": 0, "mlat": [], "tisb": [], "messages": 121, "seen": 267, "rssi": -76, "is_spoofed": true}]}

JSON format of data recorded at home-based station

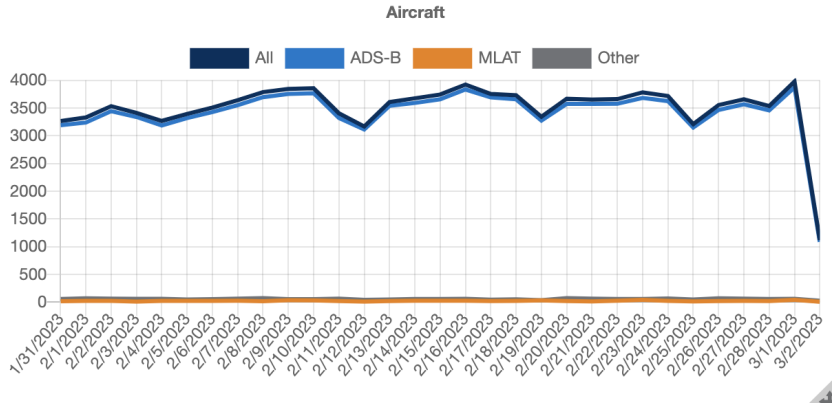
Online Data Sources

Furthermore, I used online repositories of valid ADS-B reports to feed into the Convolutional Neural network. I aggregated the data available on services such as FlightAware, Open Sky, and ADS-B Exchange by downloading their available data in a JSON format and scanning through the data and processing them into a singular CSV file to train the model.



Data display of ADS-B trailing over time on FlightAware

	3/2/2023	3/1/2023	2/28/2023	2/27/2023	2/26/2023	2/25/2023	2/24/
ADS-B Mode-S	1,087	3,874	3,458	3,570	3,466	3,149	
MLAT	8	43	21	24	19	14	
Other	26	62	58	64	70	47	
Total	1,121	3,979	3,537	3,658	3,555	3,210	



Data display of ADS-B trailing over time on FlightAware

Flight-based Data Collection

Lastly, I stowed the device to the back of a SportCruiser plane and flew with it to different locations in Los Angeles to get a wider range of ADS-B data. I flew it to different locations around Los Angeles including Santa Monica, Torrance, Malibu, Palisades, Burbank, Downtown Los Angeles, and much more. The data was logged into a CSV File and stored onto the SD Card of the Raspberry Pi for it to be used after the flight was completed.

Cockpit view of SportCruiser during flight

Getting prepared for an hour-long data collection flight over Los Angeles

Results

From the research and data collected from this experiment, I had three main takeaways for improving the detection of spoofed aircraft: aircraft telemetry indicators, RF Fingerprinting using the RSSI, and the configuration of the Convolutional Neural Network to optimize the accuracy of the model.

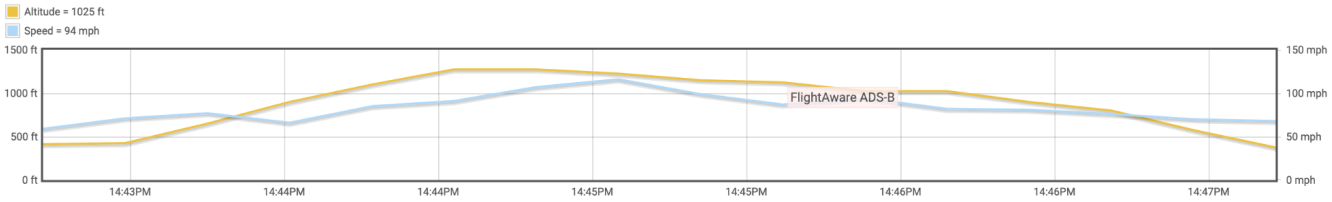
Aircraft Telemetry Indicators

Aircraft Telemetry Indicators (ATIs) which include information such as the aircraft's position, altitude, and velocity, as well as its unique identifier code can be used to detect spoofed aircraft by comparing the telemetry data received by ATIs with the expected telemetry data for the specific aircraft type. By using this technique, it is possible to detect anomalies that may indicate the presence of a spoofed aircraft (Ying, Mazer, and Bernieri, 2019). The three main indicators that signaled a spoofed aircraft in my experiment were:

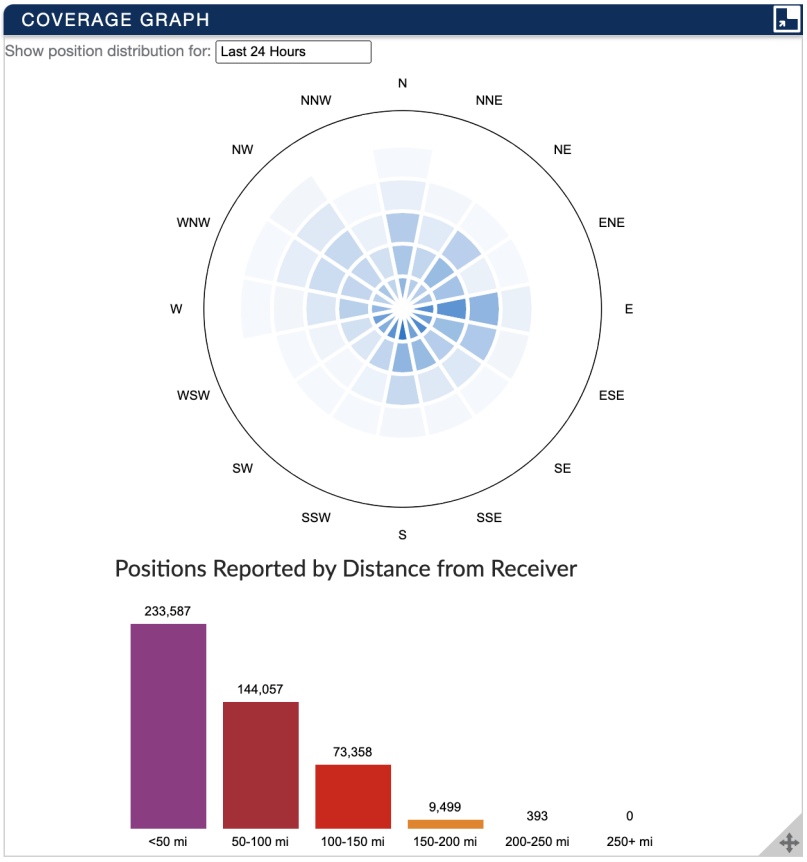
1. **Inconsistent altitude and baro rate values**: The altitude and baro rate values for each aircraft seem to vary widely and abruptly, which is not typical for genuine ADS-B data. For example, in the spoofed dataset, an aircraft has an altitude of 14981 feet and a baro rate of 51.96 feet per minute, which suggests a sudden and unrealistic change in altitude.
2. I**nconsistent or unrealistic speed data**: The ground speed values for each aircraft also seem to vary widely and abruptly, which is not typical for genuine ADS-B data.

For example, an aircraft in the spoofed dataset has a ground speed of 992 knots, which is faster than the maximum speed of most commercial aircraft.

3. **Lack of identification information**: Although the data includes flight information such as flight number, aircraft type, and registration number, there is no information about the airline or operator of the aircraft, which is typically included in genuine ADS-B data.
4. **Inconsistent or unrealistic position data**: The latitude and longitude values for each aircraft also seem to vary widely and abruptly, which is not typical for genuine ADS-B data. For example, an aircraft in the spoofed data set has a latitude of -70.63 degrees, which is beyond the southernmost point of South America which is highly unrealistic.



Altitude and Speed Graph of a Typical Genuine Aircraft



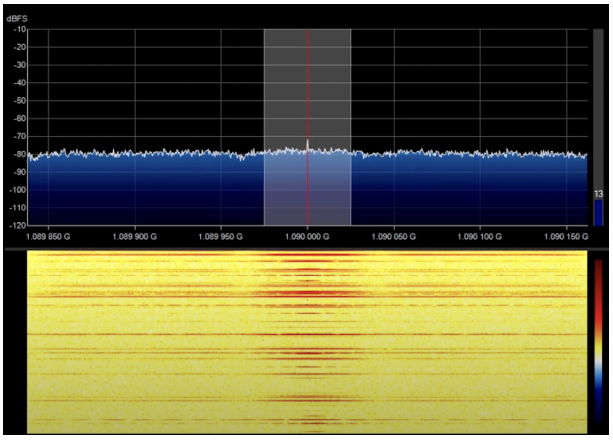
Amount of telemetry data gathered by device shown on a coverage graph

RF Fingerprinting using RSSI

RF fingerprinting using RSSI (Received Signal Strength Indicator) values of ADS-B signals can be used to detect whether an aircraft is spoofed or not by analyzing the unique RF characteristics of each transmitter. RF fingerprinting works by analyzing the variations in the signal strength of ADS-B signals received from a particular aircraft over time (Jian, Tong, and Rendon, 2023). The RF fingerprint of a legitimate aircraft is expected to be consistent, whereas the RF fingerprint of a spoofed aircraft is likely to be inconsistent due to the use of different transmitters or the use of a signal generator.

By comparing the RSSI values of the ADS-B signals received from an aircraft against a database of known RF fingerprints, it is possible to determine whether the aircraft is legitimate or spoofed. This technique can be particularly effective in detecting spoofing attacks that involve the transmission of false ADS-B data from a ground station, as the RF characteristics of the signals will be different from those of a legitimate aircraft (Haoran, Zha, and Tian, 2020).

I incorporated this into the CNN by using the RSSI values as input features, where each RSSI value corresponds to a specific time frame in the signal. Then I trained the CNN using a dataset of both spoofed and legitimate ADS-B signals to learn the unique RF fingerprint patterns associated with each type of signal. During testing, the CNN learned to classify an incoming signal as spoofed or legitimate based on its learned RF fingerprint patterns. Finally, I optimized the CNN using metrics such as precision, recall, and F1 score to improve its performance in detecting spoofed signals while minimizing false positives.



Analyzing the Frequencies of ADS-B using a RTL-SDR

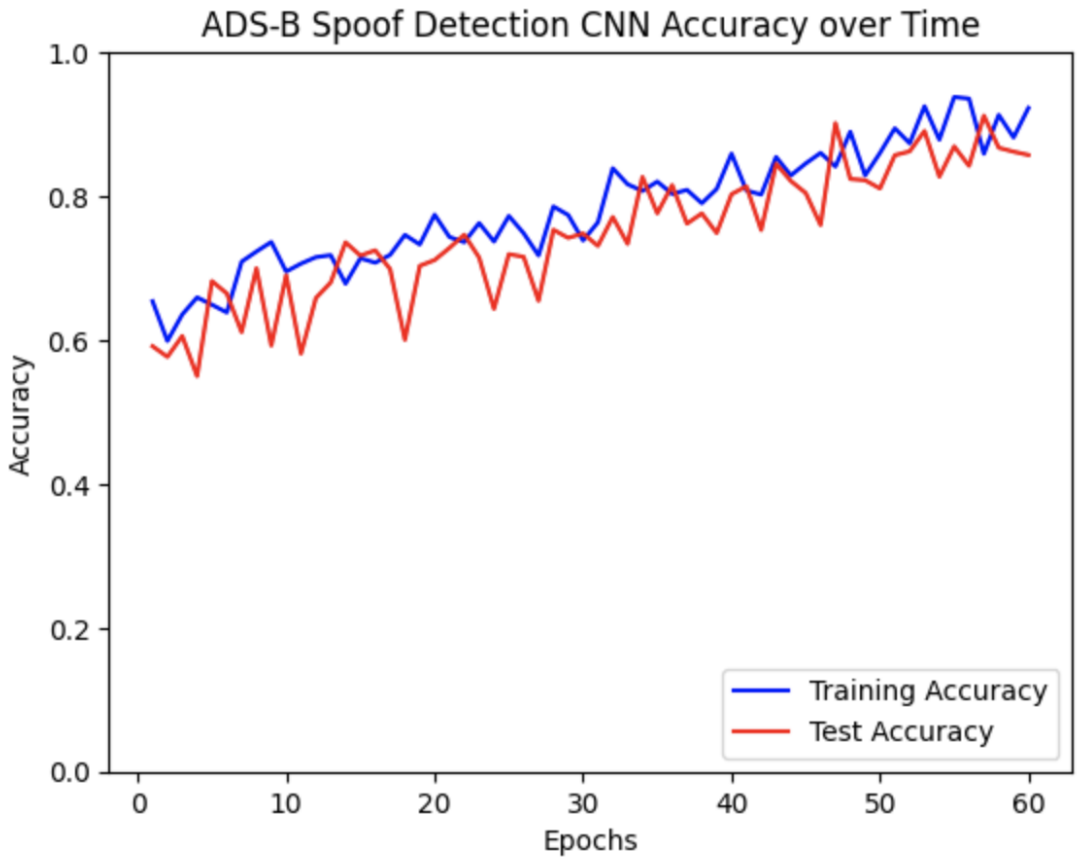
Final Output of Neural Network

I utilized a variety of techniques to fine-tune the model and improve its overall performance. One technique I utilized was to increase the complexity of the model by adding additional layers and nodes, which allowed the CNN to identify patterns and relationships within the data. I also experimented with different activation functions, including ReLU and sigmoid, to determine which worked best with the specific data set.

Another technique I employed was data augmentation, which involved generating additional training data by modifying existing data in various ways such as flipping, rotation, and zooming. This technique helped prevent overfitting and improved the CNN's ability to generalize and detect spoofed signals in new and unseen data (Amin, Clark, and Offutt, 2014).

Finally, I utilized transfer learning, which involved taking a pre-trained CNN and fine-tuning it for the specific task of detecting spoofed ADS-B signals. This approach allowed me to leverage the knowledge and expertise that had already been built into the pre-trained model and apply it to the new task at hand.

Overall, through a combination of these techniques, I was able to significantly improve the accuracy of the CNN, ultimately achieving a level of over 90% accuracy by 60 epochs.



Graph of CNN Accuracy Performance

Discussion

The results of this experiment indicate that there are several effective techniques for improving the detection of spoofed aircraft in ADS-B signals. By analyzing the aircraft telemetry indicators, it was possible to detect anomalies that may indicate the presence of a spoofed aircraft. The use of RF fingerprinting using RSSI values also proved to be an effective technique for detecting spoofed signals, particularly in cases where a ground station is transmitting false data.

Incorporating these techniques into the CNN required careful consideration and optimization of various parameters such as the model complexity, activation functions, and data augmentation. Increasing the model complexity and experimenting with different activation functions helped the CNN identify patterns and relationships within the data more effectively. Data augmentation was also useful in generating additional training data and preventing overfitting. Through a combination of these techniques, the CNN was able to achieve a high level of accuracy, ultimately reaching more than 90% by 60 epochs.

In terms of future expansion of the project, there are multiple avenues that could be pursued. One potential area of focus could be to refine the detection system to improve its ability to differentiate between different types of spoofing attacks. For example, some

spoofing attacks involve the transmission of false GPS signals to manipulate an aircraft's position, while others involve the use of false ADS-B signals to mimic the behavior of a legitimate aircraft. Developing more sophisticated detection techniques for different types of spoofing attacks could further enhance aviation safety.

Another potential area for future expansion is to explore the use of machine learning techniques beyond CNNs for detecting spoofed aircraft. For instance, deep learning techniques such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks could be used to improve the detection accuracy of spoofed aircraft. Additionally, reinforcement learning techniques could be used to develop more adaptive and robust detection systems that can learn and adapt to new types of spoofing attacks over time.

Conclusion

In conclusion, this study conducted on improving the detection of spoofed aircraft using aircraft telemetry indicators, RF fingerprinting using RSSI, and the configuration of the Convolutional Neural Network showed promising results. The use of aircraft telemetry indicators proved useful in detecting spoofed aircraft by comparing the telemetry data received by ATIs with the expected telemetry data for the specific aircraft type. Additionally, RF fingerprinting using RSSI values of ADS-B signals provided a unique method to detect spoofed aircraft by analyzing the unique RF characteristics of each transmitter. The utilization of the Convolutional Neural Network aided in the classification of incoming signals as spoofed or legitimate based on the learned RF fingerprint patterns. Finally, a combination of techniques such as increasing the complexity of the model, data augmentation, and transfer learning were used to fine-tune the CNN and improve its overall performance.

Acknowledgments

I would like to express my gratitude to those who have helped me in creating and developing Fly Catcher and conducting research. First and foremost, I would like to thank my flight instructor for his guidance and support throughout this project. I also want to thank my science teacher and mentor for their insights and encouragement about the computer science theory and physics behind the device. Additionally, I am grateful to my physics teacher and flight school for providing me with the resources necessary to create this device.

Additionally, I would like to extend my appreciation to the cybersecurity researchers who gave me feedback and valuable suggestions to improve the neural network and the device. As well as special thanks to the fellow pilots at my flight school and air traffic control personnel who shared their personal advice on the device and how it can be improved.

Finally, I would like to thank all the individuals who supported me throughout this project, including my family, friends, and fellow classmates. Their encouragement and support have been critical in making this project a success.

Bibliography

Flight tracker / flight status. FlightAware. (n.d.). Retrieved March 2, 2023, from https://flightaware.com/

Haoran, Zha & Tian, Qiao & Lin, Yun. 2020. Real-World ADS-B signal recognition based on Radio Frequency Fingerprinting. 1-6

Home - serving the flight tracking enthusiast - ADS-B exchange. ADS. (2023, January 26). Retrieved March 2, 2023, from https://www.adsbexchange.com/

S. Amin, T. Clark, R. Offutt and K. Serenko. 2014. Design of a cyber security framework for ADS-B based surveillance systems. Systems and Information Engineering Design Symposium (SIEDS) Charlottesville, VA, USA. 304-309

Jing Wang, Yunkai Zou, and Jianli Ding. 2020. ADS-B spoofing attack detection method based on LSTM. EURASIP J. Wirel. Commun. Netw. 2020, 1 (Nov 2020). https://doi.org/10.1186/s13638-020-01756-8

Jian, Tong & Rendon, Bruno & Ojuba, Emmanuel & Soltani, Nasim & Wang, Zifeng & Sankhe, Kunal & Gritsenko, Andrey & Dy, Jennifer & Chowdhury, Kaushik & Ioannidis, Stratis. (2020). Deep Learning for RF Fingerprinting: A Massive Experimental Study. IEEE Internet of Things Magazine. 3. 50-57. 10.1109/IOTM.0001.1900065.

Meides, M. (n.d.). *Network opensky explorer emergency alerts coverage & facts receiver ranking VHF/voice feeding*. The OpenSky Network. Retrieved March 2, 2023, from https://opensky-network.org/

Ying, X., Mazer, J., Bernieri, G., Conti, M., Bushnell, L., & Poovendran, R. (2019). Detecting ADS-B Spoofing Attacks Using Deep Neural Networks. *2019 IEEE Conference on Communications and Network Security (CNS)*, 187-195.