ChatGPT

Plan for AI-Based RF Interference (GNSS Jamming) Detection and Mitigation

1. GNSS Jamming Fundamentals and Motivation

Figure: Main principle of GNSS spoofing vs. jamming attacks – a spoofer (left) transmits counterfeit satellite signals to mislead the receiver, whereas a jammer (right) emits strong RF noise on the GNSS frequency to overwhelm genuine signals [1].

GNSS Signal Vulnerability: GNSS satellites transmit from ~20,000 km altitude, so by the time their signals reach Earth, the signal power is extremely weak – often near the thermal noise floor [2]. This makes GNSS receivers **highly susceptible** to interference. A malicious transmitter ("jammer") broadcasting even a modest RF signal on the GNSS frequency can drown out the authentic satellite signals. Studies show that an interference source of only ~1 W can disrupt GNSS receivers within a radius of ~15 km [3], underscoring how easily GNSS service can be knocked out by a jammer. Given modern society's heavy reliance on GNSS for navigation, timing, and communications synchronization, successful jamming can cause **significant disruption** to transportation, military operations, and critical infrastructure [4] [5].

What is Jamming? In formal terms, *jamming* is *"deliberate RF interference caused by emissions intended to render a wanted signal unintelligible or unusable"* [6]. Essentially, the jammer transmits a powerful signal (often just broadband noise or a simple tone) on or near the GNSS frequency band, **overpowering the faint GNSS signals** so the receiver can no longer lock onto the satellites [6]. This is distinct from spoofing (which broadcasts fake GNSS-like signals to deceive the receiver with false coordinates). Jamming comes in various forms – common types of intentional GNSS interference include continuous-wave (CW) tones, narrowband carriers, pulsed signals, chirp sweeps, and broadband noise jammers [7]. Each type can disrupt GNSS operation by raising the noise floor or saturating the receiver's front-end.

Real-World Relevance: The threat of GNSS jamming is no longer theoretical. Low-cost, off-the-shelf hardware (cheap SDR platforms like USRPs or HackRF) and online instructions have made jamming devices widely accessible [8] [9]. **Recent conflicts and incidents** highlight the prevalence of GNSS jamming. For example, during military exercises in 2018, northern Finland experienced large-scale GPS disruptions attributed to jamming [10]. More recently, since the 2022 Russia-Ukraine conflict, multiple countries near Russia have reported a sharp increase in GNSS interference affecting aviation and maritime navigation [11]. These events underscore the urgent need for robust jamming **detection and mitigation** – early detection is essential so that countermeasures can be activated to avoid service degradation [9]. In summary, GNSS jamming is an important and timely problem to tackle, motivating our project's focus on AI-based detection and mitigation techniques.

2. I/Q Data and Signal Processing for Interference Detection

Understanding I/Q Data: To detect RF interference like jamming, we will work with **in-phase/quadrature (I/Q) signal data** captured from the GNSS RF front-end. I/Q data is the raw complex digital signal (after down-conversion and sampling) containing the full information of the RF waveform – essentially how the signal's amplitude and phase evolve over time. By analyzing I/Q samples, one can observe the presence of abnormal interference patterns that would not be evident from higher-level measurements alone. For example, a strong broadband jammer will manifest as a sudden surge in the power of the I/Q samples (since noise floor rises), while a continuous-wave jammer might appear as a spike at a particular frequency in the Fourier transform of the I/Q stream. Using I/Q data gives us the **highest-fidelity view** of the signal, which is crucial for training a Deep Neural Network (DNN) to recognize subtle interference signatures.

Collecting and Preprocessing Data: A key early task is to obtain labeled I/Q data for both "normal" GNSS reception and "jammed" conditions. We will discuss data collection with our supervisor, but likely approaches include:

- *Controlled SDR-Based Experiments:* One option (as used in recent research) is to set up a software-defined radio to **simulate jamming signals** while recording GNSS signals. For instance, Sormayli *et al.* (2025) conducted experiments using an SDR transmitter to generate jamming while a Ublox GNSS receiver collected the I/Q data under those interference scenarios [12]. We could recreate a similar setup: e.g. use a GNSS signal simulator or live sky GNSS reception combined with an SDR broadcasting a jamming waveform (carefully confined to a lab or shielded environment for legality). This would yield ground-truth datasets of I/Q samples with known interference types and power levels.

- *Public Datasets or Simulation:* We will also check for any existing datasets. While several public datasets exist for GNSS *spoofing* (e.g. TEXBAT and OAKBAT for spoofing/interference scenarios [13]), pure jamming datasets are less common. If needed, we can simulate GNSS signals and jamming in software (using Python libraries or GNSS SDR simulators) to generate training data. The advantage of simulation is the ease of labeling and variety (e.g. simulate CW jamming, chirp jamming, etc. at different Jamming-to-Noise Ratios or JNRs). The disadvantage is that simulated data may not capture all real-world effects (multipath, hardware impairments), so a mix of real and simulated data could be ideal.

Before feeding the I/Q data to any ML model, we will perform **signal preprocessing** and feature extraction:

- We may apply basic filtering or down-sampling if needed (for example, isolating the GPS L1 band and decimating to manageable sample rates).
- The continuous I/Q stream will be segmented into time-windowed samples (since a DNN or classifier will likely operate on fixed-length input segments, e.g. a few milliseconds of data).

- For classical machine learning approaches, we will extract informative features from each segment. Traditional interference detectors look at metrics like Automatic Gain Control (AGC) levels, signal power, noise variance, and frequency-domain peaks [14]. A sudden jump in AGC or noise floor is a strong indicator of jamming [14]. We can compute features such as: mean and variance of signal power, FFT power spectral density (to spot narrowband spikes), or correlation metrics from the GNSS receiver (drop in C/N_0, etc.). These features can feed a machine learning classifier. In the cited real-

time study, the I/Q data was pre-processed with normalization, correlation analysis, and feature selection to pick the most relevant indicators before training an XGBoost classifier [12]. We can follow a similar approach for an initial baseline model.

- For deep learning approaches, we might bypass manual feature extraction and feed the raw I/Q (or a transformed representation) directly into a neural network. One effective technique from recent literature is converting I/Q data into **2D time-frequency images** – for example, compute spectrograms or Fourier transforms of the signal over short windows – and then use Convolutional Neural Networks on those images [15] [16]. We will consider creating such representations (e.g. spectrogram or even simpler, the power spectrum or histogram of I/Q samples) as inputs to a CNN. This leverages CNNs' strength in pattern recognition (treating interference detection somewhat like an image recognition problem on the signal's spectral signature). In fact, Chen *et al.* (2025) found that using time-frequency spectrogram images yielded excellent performance in jamming detection – their ResNet-18 model could detect interference with 90% accuracy even at JNR = –20 dB (extremely low jammer power relative to noise) [16].

Overall, assembling a high-quality I/Q dataset and applying appropriate preprocessing is a critical foundation. We will aim to gather a diverse set of scenarios (no interference vs. various jammer types and strengths) to train a robust model. If needed, we'll begin with simulated data to develop our pipeline, and later incorporate any real-world captures we can obtain (to improve realism).

3. Machine Learning Approaches for Jamming Detection and Mitigation

Traditional Detection vs. ML: Historically, GNSS interference detection relied on signal processing algorithms with fixed thresholds or statistical tests. Common methods include monitoring the AGC level or C/N_0 for sudden drops, checking for anomalies in the correlation peaks of the receiver, or using FFT analysis to spot unexpected strong spectral lines [14] [17]. While these methods are simple and fast, they can struggle to distinguish deliberate jamming from natural signal fades or incidental RF noise, and they require manual threshold tuning [18] [19]. This has led to growing interest in **machine learning** techniques that can learn to recognize jamming patterns from data. In fact, surveys report that in recent years ML-based methods have become *"the most popular methods for detecting GNSS jamming and spoofing attacks"* [20].

Supervised ML Classification: A straightforward approach is to treat interference detection as a binary (or multi-class) classification problem: given some representation of the signal, output whether it's "clean" or contains a jammer (and possibly identify the jamming type). We will explore several ML models for this task:

- *Baseline Algorithms:* We can start with classical algorithms like **K-Nearest Neighbors (KNN)**, **Support Vector Machines (SVM)**, or **Decision Trees/Random Forests**, using features extracted from the I/Q data. These models have been tested in literature. For example, KNN was applied to classify chirp jamming in GNSS signals at the pre-correlation stage [21], and SVMs have shown strong performance in detecting spoofing/jamming in prior studies [22]. A recent comparison even found decision tree models (e.g. XGBoost) to be very effective for GNSS interference classification [23]. One 2025 experiment implemented an **XGBoost classifier** on an embedded microcontroller and achieved a 99.97% jamming detection rate (with 99.94% precision) on a test dataset, while keeping inference

time to only 20 µs per sample – demonstrating that lightweight ML can work in real time [24]. We can use such algorithms as a starting point to get a baseline accuracy on our data.

- *Deep Neural Networks:* The core of our project is to leverage a Deep Neural Network on raw I/Q data. Deep learning can potentially capture more complex patterns of interference than manual features. We will consider architectures like **Multi-Layer Perceptrons (MLP)**, **Convolutional Neural Networks (CNNs)**, and **Recurrent Neural Networks (RNN/LSTM)**:

- *MLP on Features:* As an intermediate step, a fully-connected neural network can be trained on the same feature set as above (AGC, power metrics, etc.) to see if it learns non-linear combinations of features better than, say, SVM. MLPs have indeed been used: one study trained a multi-layer perceptron to classify various jammer types and reported ~98.9% detection accuracy [25].
- *CNN on Raw or Images:* CNNs can directly ingest raw time-series or 2D transforms. Prior work suggests two promising CNN approaches: (1) feeding the raw I/Q sequence into a 1D CNN (or into an LSTM) to learn temporal patterns, or (2) feeding a time-frequency image into a 2D CNN. The image-based approach has been very successful: researchers converted jamming signals into spectrograms, power spectral plots, or even sample histograms, and then applied deep CNN or transfer learning models to classify interference [26]. For instance, using ResNet-18 on spectrogram images not only achieved high detection rates but could also **identify** the type of interference (CW vs. chirp vs. pulse, etc.) at high accuracy [15] [16]. We might replicate a similar approach by generating spectrogram images from our I/Q data and training a CNN to detect jamming. This can be done in Python using libraries like TensorFlow/PyTorch for the model and matplotlib or scipy for signal processing.
- *RNN/LSTM:* Recurrent networks (LSTM or GRU) can be effective for sequence data like I/Q samples. In literature, LSTM networks have been used to detect weak GNSS signals under jamming, by learning temporal dependencies in the signal sequence [27]. A bidirectional LSTM model achieved over 98% accuracy in detecting interference in one study [28]. We may experiment with an LSTM-based classifier on raw I/Q sequences for comparison with the CNN approach.

By reviewing these methods, our plan is to **implement and compare** a couple of models. We might start with a simpler classifier (like XGBoost or SVM) using a small set of features to establish a baseline performance on our data. Then, we will move to a deep learning model (likely a CNN on spectrograms, given the strong results reported) to see if we can improve detection sensitivity or robustness. Key performance metrics will include detection accuracy, false alarm rate, and the lowest JNR at which the model can still reliably flag the jammer (since a good detector should catch even low-power interference).

Interference Mitigation Strategies: Detection is the first step; mitigation is the ultimate goal of an anti-jamming system. Once our model can detect a jamming event, we will consider techniques to *mitigate* the interference's impact on the GNSS receiver:

- *Traditional Mitigation:* Depending on the jammer type, different signal processing countermeasures can be applied. For example, for a narrowband continuous wave jammer, one can apply a notch filter or frequency-domain **pulse blanking** to remove that frequency component [29]. For pulsed jammers, **time-domain blanking** (zeroing out samples during interference pulses) is effective [29]. Chirp jammers (swept-frequency) are trickier, but since they are sparse in time-frequency, adaptive filtering or time-frequency excision can be used [29]. In our project scope, if we identify the presence (and perhaps rough type) of jamming, we could implement a simple mitigation like: *ignore or down-weight*

jammed signals (e.g. drop the affected frequency bins, or discard measurements from satellites that are degraded).

- *AI-Assisted Mitigation:* A more advanced concept, beyond just detection, is using AI to choose the optimal mitigation technique in real-time. One 2025 study introduced an "intelligent interference mitigation" system where a deep learning model examines the interference and automatically selects the best mitigation approach (filtering, blanking, etc.) to maximize the receiver's signal quality [30]. Their method improved the GNSS receiver's carrier-to-noise ratio by over 10% compared to traditional fixed mitigation methods [31]. While developing a full AI-driven mitigation system might be beyond our initial scope, we will keep this in mind. It suggests a possible extension: our DNN could be trained not only to detect jamming but also to classify the type of interference (CW vs. chirp vs. pulse). With that classification, we could then apply a targeted countermeasure. For example, if the model recognizes a narrowband CW jammer, we activate a notch filter at the identified frequency; if it detects a wideband noise jammer, perhaps switch to a robust tracking mode or use antenna nulling if available.

In summary, the literature review indicates that a variety of ML techniques (from simple KNN/SVM to deep CNNs) have been successfully used for GNSS jamming detection, often achieving high accuracy ($\geq$98–99% in controlled tests) [25] [24]. Deep learning in particular offers strong detection capabilities even under low JNR conditions [16]. Building on these insights, our plan is to implement a DNN-based detector using Python, train it on I/Q data, and then integrate it into a prototype system that can both raise alerts and trigger interference mitigation actions. By doing so, we hope to significantly improve the resilience of GNSS receivers against jamming attacks.

4. Proposed Project Plan and Next Steps

To organize our efforts, here is a **step-by-step plan** for the project:

1. **In-Depth Literature Review:** Begin with gathering and studying resources on the three key knowledge areas – GNSS jamming fundamentals, I/Q signal processing, and machine learning techniques for RF interference. (The findings summarized above provide a starting point. [9] [21]) We will continue to read survey papers and recent research (e.g. the *Sensors* 2024 review [32], and 2025 papers [21] [15]) to solidify our understanding and identify state-of-the-art methods.

2. **Data Acquisition Strategy:** In parallel, determine how to obtain training data:

3. Coordinate with our supervisor about hardware availability (GNSS receivers, antennas, SDRs). If possible, set up a controlled experiment to collect real I/Q data with an SDR-generated jammer [12]. Ensure compliance with regulations (likely use shielding or dummy loads to avoid transmitting interference openly).
4. If hardware trials are delayed, start with *simulated data*: use Python libraries to generate a pseudo-GNSS signal (or use recordings of actual GPS signals) and add synthetic jamming (noise or CW interference) at various power levels. This simulated dataset can be used to develop our ML models initially.

5. Explore any open datasets (e.g. check if OAKBAT includes jamming recordings, or if any researchers are willing to share data).

6. **Data Preparation and Exploration:** Once data is available, perform preprocessing:

7. Convert raw capture files into manageable chunks of I/Q samples. Label each chunk as “clean” or “jammed” (or by jammer type if applicable).
8. Visualize some examples: plot time-domain waveforms and spectra to see the differences when jamming is present (this will also help us decide on good input representations for the DNN).

9. Compute candidate features for classical ML (e.g. power, variance, FFT peaks, etc.) and see which ones show clear separation between jammed vs non-jammed cases.

10. **Baseline ML Model:** Implement a simple classifier on the extracted features (for instance, train an XGBoost or SVM using scikit-learn). Evaluate its performance on a held-out test set. This will give us a baseline accuracy to improve upon and also help validate that our features and labels make sense (e.g. if even a simple model gets high accuracy, it means the jamming signature is learnable from the data).

11. **Develop DNN Model:** Move on to the deep learning approach:

12. Decide on an input format for the DNN. For example, generate spectrogram images from the I/Q data (using a sliding window STFT) and use a CNN, or feed a sequence of raw I/Q values into an LSTM/CNN hybrid model. Based on the literature, we lean towards a 2D CNN on time-frequency images as a promising route [15] [26].
13. Build and train the model in Python (using PyTorch or TensorFlow). Start with a known architecture (perhaps ResNet-18 or a smaller CNN if resources are limited) and train it on our dataset. Monitor training/validation accuracy and adjust as needed (tuning hyperparameters, data augmentation, etc.).

14. Evaluate the trained DNN on test data. Compare its detection accuracy and false alarm rate against the baseline model. Also test its sensitivity: for instance, test on low JNR cases to see if it can detect weaker jamming better than the baseline.

15. **Interference Mitigation Integration:** After achieving reliable detection, implement a rudimentary mitigation in our testbed:

16. For example, if our system flags a jammed condition, we can simulate the receiver response by blanking out the affected signal portion. In practice, this could mean not using the affected frequency or satellite signal in position calculation. We can demonstrate this by measuring the improvement in C/N_0 or position solution once the mitigation is applied. (If time permits, this step can be expanded by incorporating an ML decision on which mitigation to apply, as discussed in Section 3.)

17. Evaluate the effectiveness of mitigation (e.g. does applying a notch filter when a CW jammer is detected improve the C/N_0 by the expected amount? Does the system maintain a GPS fix under jamming with the mitigation, versus losing the fix without it?).

18. **Iteration and Refinement:** Throughout the implementation, iterate based on results:

19. If the ML model isn't performing well, revisit feature engineering or try a different architecture (e.g. if CNN on spectrogram isn't yielding >90% accuracy, try an LSTM on raw data, or add more training examples, etc.).
20. If data is insufficient, generate or collect more (perhaps simulate additional scenarios or augment data by adding noise, shifting frequencies, etc., to generalize the model).

21. Optimize the model for real-time use if needed (e.g. the final model might be simplified or compressed so it could run on an embedded device, inspired by techniques like the LcxNet-Fusion network which reduced model size by 43% for faster inference [33] [34]).

22. **Documentation and Analysis:** Keep detailed notes of literature findings, data collection procedures, and modeling results. The final report will include a literature review (for which this plan and collected references will be useful), a methodology section describing our data and models, and an evaluation of results. We will also discuss limitations and potential future extensions (for example, extending to GNSS spoofing detection, or using the system in a live GNSS receiver setup).

By following this plan, we will first build a strong foundation of domain knowledge, then acquire the necessary data, and progressively develop an AI-based solution for GNSS jamming detection. The combination of a thorough literature review and an iterative experimental approach will put us in a good position to deliver a successful semester project on RF interference detection and mitigation.

Overall, the project promises to deepen our understanding of GNSS signal vulnerabilities and showcase how modern machine learning (especially deep learning) can enhance interference detection capabilities beyond what traditional methods offer [20] [28]. With GNSS jamming on the rise globally, our work will be tackling a very relevant problem, and the skills/knowledge gained (software-defined radio, signal processing, neural networks) will be valuable going forward. Let's get started!

---

[1] [6] [8] [9] [13] [18] [19] [20] [22] [23] [25] [32] Recent Advances on Jamming and Spoofing Detection in GNSS
https://www.mdpi.com/1424-8220/24/13/4210

[2] [4] [5] [12] [14] [17] [21] [24] [26] [27] [28] Real-Time jamming detection using windowing and hybrid machine learning models for pre-saturation alerts | Scientific Reports
https://www.nature.com/articles/s41598-025-10567-0?error=cookies_not_supported&code=5f57019e-1770-4b2d-9354-9b9639759cd3

[3] [7] [29] [30] [31] Frontiers | GNSS interference mitigation method based on deep learning
https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2025.1535906/full

[10] [15] [16] [33] [34] Frontiers | Deep learning-based GNSS composite jamming detection and recognition technology
https://www.frontiersin.org/journals/signal-processing/articles/10.3389/frsip.2025.1567926/full

[11] Russian electronic warfare base linked to GPS jamming across Baltic region - GPS World
https://www.gpsworld.com/russian-electronic-warfare-base-linked-to-gps-jamming-across-baltic-region/