# Predictive Modeling of Frequency Hopping Signals via Deep Reinforcement Learning

Van Dat Tuong
*School of Information and Communications Technology*
*Hanoi University of Science and Technology*
Hanoi, Vietnam
dat.tuongvan@hust.edu.vn

Khanh Nguyen-Trong
*Falculty of Information Technology 1*
*Posts and Telecommunications Institute of Technology*
Hanoi, Vietnam
khanhnt@ptit.edu.vn

Nhu-Ngoc Dao
*Department of Computer Science and Engineering*
*Sejong University*
Seoul, Republic of Korea
nndao@sejong.ac.kr

Sungrae Cho
*School of Computer Science and Engineering*
*Chung-Ang University*
Seoul, Republic of Korea
srcho@cau.ac.kr

***Abstract*—With the rapid expansion of legal and illegal activities using unmanned aerial vehicles (UAVs), the UAV incidents in the unarmed population are gradually increasing. To prevent the damage from such UAV incidents for the public facility, jamming over the communication between UAVs and their remote control is a cost-effective solution. However, modern UAVs are enabled to inherently reduce susceptibility to interference. Specifically, they are installed with frequency hopping signals that make a huge challenge for efficient jamming owing to the narrowband transmissions and rapid frequency changes. Therefore, it is necessary to develop an agile model that precisely and rapidly predicts parameters of frequency hopping signals to improve the jamming efficiency. This study presents a deep reinforcement learning framework capable of predicting future time-frequency locations of the signals. In particular, a deep deterministic policy gradient (DDPG) algorithm is developed and trained on empirical signal data collected from drone DJI Mavic 3 Pro. A novel reward function is introduced that evaluates prediction accuracy by combining the results of time and frequency predictions. Training results indicate a high fidelity of signal forecasting, achieving an excellent mean score that significantly converges with low standard deviation.**

***Index Terms*—Frequency hopping, reinforcement learning, unmanned aerial vehicles.**

## I. Introduction

Unmanned aerial vehicles (UAVs) have become a dominant category of unmanned systems, with numerous applications in various areas such as search and rescue, logistics, and environmental monitoring [1]–[3]. Unforunately, the increasing of UAV incidents in public areas motivates jamming solutions. While UAVs are typically and remotely operated, they depend on a wireless link with command and control (C2) signals that are also known as frequency hopping signals, most commonly within the 400MHz, 900MHz, 1.2-1.3GHz, 2.4GHz, or 5.8GHz bands, for real-time communication with ground control stations [4]. Targeted radio frequency (RF) jamming can disrupt the UAVs C2 link, potentially forcing them into a fail-safe mode such as hovering, returning to home, or landing. In particular, jamming efficiency depends on accurate detection and prediction of C2 frequency hopping signals.

Modern UAVs increasingly incorporate adaptive communication strategies to evade detection and maintain link robustness. Specifically, UAVs vary their hopping signals or adapt their modulation in response to environmental interference or jamming attempts, present a complex and evolving target. Therefore, efficient jamming requires the development of an agile model capable of identifying frequency hopping signals in real-time, even in the presence of dynamic and unpredictable hopping behavior. These algorithms must not only detect the presence of the signals but also infer or predict their future behavior to enable precise and timely jamming.

While a substantial body of research exists on the detection of radio-controlled UAVs [5], [6], relatively little attention has been devoted to the predictive models of UAVs frequency hopping signals. This gap is particularly significant in the context of UAV RF jamming, where the ability to anticipate and disrupt UAV C2 links is critical. Existing approaches to signal identification typically rely on extended observation periods, requiring prolonged signal monitoring before any predictive inference can be made. Although several recent work has demonstrated the feasibility of recognizing the signals within several hopping interval [7], [8], these methods still suffer from latency that may limit their operational applicability in real-time jamming scenarios.

This study proposes a novel and efficient framework for predicting the near-future evolution of frequency hopping signals using only a short and fixed observation window. Our approach leverages deep reinforcement learning (DRL) to learn time-frequency evolution and forecast the subsequent behavior of the signals. The proposed framework not only enables accurate prediction of the signal's spectral trajectory but also offers tactical flexibility: jamming systems either avoid interfering with legitimate signals or, conversely, selectively specific target

signals for disruption. This predictive capability enhances the precision and responsiveness of jamming strategies, supporting real-time decision-making in rapidly evolving electromagnetic environments. To achieve our goal, we proceed as follows. First, we construct the dataset as the RF signals of drone DJI Mavic 3 Pro and conduct data preprocessing, as shown in section II. Second, we design the training algorithm in section III. Finally, we evaluate the trained model in section IV, the model performance on the test dataset. The major contribution of this work can be summarized as follows:

- We introduced a state representation of frequency hopping signals, which was utilized in training a DRL model designed to predict the near future evolution of the signals, only based on a small observation window.
- We developed a DRL model for training the frequency hopping prediction based on Deep Deterministic Policy Gradient (DDPG) algorithm. In particular, a reward function was designed that combines the accuracy when predicting time and frequency. This reward plays as a metric to assess the prediction accuracy.
- We conducted simulations to demonstrate the prediction results considering frequency hopping signals of the drone DJI Mavic 3 Pro.

## II. Dataset Construction

### A. Data Collection

The RF signal data used in this study is collected from a drone DJI Mavic 3 Pro, which employs frequency hopping for its C2 communication link. Signal measurements are conducted across multiple experimental runs to ensure consistency and reproducibility. Each measurement session is captured in approximately 5 seconds of RF data using a software-defined radio (SDR) testbed in a controlled laboratory environment. The SDR system was configured with a measurement bandwidth of 200MHz, centered at 2.4GHz, corresponding to the ISM band where the C2 link operates. To capture the temporal dynamics of the transmissions, RF snapshots are recorded at a repetition interval of 100 $\mu$s, with 1024 complex I/Q samples acquired per snapshot. These settings are selected to balance temporal resolution with signal fidelity, enabling accurate reconstruction of the hopping behavior. It is noted that the laboratory environment introduces occasional background noise, which is both expected and considered beneficial for the study. The presence of such noise contributes to the realism of the dataset and supports the evaluation of the proposed predictive model under non-ideal, interference-prone conditions, as typically encountered in real-world UAV jamming scenarios.

Drone DJI Mavic 3 Pro is evaluated under multiple operating modes across five independent measurement runs. Each run is conducted to assess potential variability in the emitted frequency hopping signal with respect to the drone's operational state. However, analysis of the captured data revealed no discernible differences in the signal characteristics across different modes. Consequently, no differentiation is made between individual measurement runs or operational configurations in subsequent analysis. All measurement runs are performed that considers ambient RF activity. These background signals are intentionally retained in the dataset and incorporated into both the training and test datasets for two key reasons: (i) to enable the model to learn to discriminate between true signal presence and background noise, thereby reducing the risk of false-positive predictions during inference; and (ii) to provide a realistic performance baseline, reflecting the model's ability to operate under conditions that approximate real-world spectral environments. This methodology promotes generalization beyond idealized signal conditions, thereby enhancing the robustness and reliability of UAV signal detection and prediction models under realistic and variable electromagnetic environments.

TABLE I
Block Descriptor Words (BDWs)

| No | Data field | Data size | Unit | Note |
|---|---|---|---|---|
| 1 | bid | 1x1 | N/A | Block ID |
| 2 | sid | 1x1 | N/A | Antenna ID |
| 3 | toa | 1x1 | ns | Time of arrival |
| 4 | tod | 1x1 | ns | Time of departure |
| 5 | fs | 1x1 | Hz | Sampling frequency in wide band |
| 6 | fb | 1x1 | Hz | Sampling frequency in narrow band |
| 7 | fd | 1x1 | Hz | Sampling frequency for PC |
| 8 | fc | 1x1 | Hz | Center frequency |
| 9 | bw | 1x1 | Hz | Bandwidth |
| 10 | amp | 2x1 | N/A | Amplitude |
| 11 | nxx | 1x1 | N/A | Estimated noise background |
| 12 | xiq | 3x1 | N/A | IQ signal (bounding and phase) |
| 13 | rsv | 1x1 | N/A | Buffer |

### B. Data Preprocessing

First, the collected raw I/Q data is preprocessed to estimate basic information of the signal, which is called Block Descriptor Words (BDWs), as described in Table I. We assume BDWs are estimated at low layer and available for further processing. The original BDWs are reloaded using the python SciPy library, wherein the information is structured as key-value pairs. The keys are string identifiers that represent various data attributes, including: 'fc', 'bw', 'amp', 'snr', 'fs', 'fd', 'nxx', and 'dp'. The corresponding values for each attribute are reloaded and subsequently normalized according to the principles outlined below:

- The BDW data is reloaded and concatenated in the appropriate temporal sequence as originally collected. The BDW sequence is segmented at data points where a change in center frequency or bandwidth is detected, indicating a frequency hopping event. Each segment, referred to as a sub-BDW sequence, represents a continuous signal state defined by a fixed pair of center frequency and bandwidth values.
- The center frequency is characterized using two normalized descriptors: (i) Frequency Band Index, for instance,

in the context of UAV signal analysis, four primary operating bands are considered such as 400MHz, 900MHz, 2.4GHz, and 5.8GHz, assigned indices of 0, 0.25, 0.5, and 1.0, respectively; and (ii) Normalized Center Frequency, within each band, the center frequency value is scaled to a [0, 1] range relative to the minimum and maximum values of the respective frequency band.

- The signal bandwidth is normalized to the [0, 1] range based on the maximum observed channel bandwidth. For example, in the case of drone DJI Mavic 3 Pro, the maximum bandwidth is identified as 40MHz.
- For each of the remaining features ('amp', 'snr', 'fs', 'fd', 'nxx', 'dp'), the normalized representation includes two statistics: (i) the mean value, normalized to the [0, 1] range using the maximum value observed within the current sub-BDW sequence; and (ii) the standard deviation, calculated within the same sub-BDW context.
- To capture the temporal dynamics of each sub-BDW sequence, three additional attributes are included: (i) 'ts': the time of arrival (toa) of the first BDW in the sequence; (ii) 'te': the time of departure (tod) of the last BDW in the sequence; and (iii) 'nb': the total number of BDWs within the sub-sequence.

This data preprocessing procedure effectively constructs a temporal profile of signal state transitions, capturing variations in center frequency, channel bandwidth, and associated signal characteristics.

## III. Training Algorithm

### A. Frequency Hopping System Modeling

The frequency hopping system model is developed using the python Gym library, a framework that facilitates the construction of simulation environments for RL. This library provides extensive support for rapid information transformation and includes modules for object visualization, making it well-suited for modeling dynamic systems. In this model, signal behavior is constructed based on the collected BDWs. The training process continuously evaluates the effectiveness of predicting frequency hopping parameters using a reward function, which quantifies prediction accuracy by measuring the deviation between predicted parameters and actual observations. This reward function guides the adjustment of weights within the deep neural networks to maximize the prediction accuracy.

The design of the frequency hopping model incorporates the following key considerations:

- Accurate signal state transitions: To ensure realism, the model must fully replicate the signal state changes observed in the BDW dataset. This requires that all signal-related attributes accurately reflect the recorded data, including the historical transitions of each signal state feature.
- Incorporation of noise to address data limitation: Since the available dataset may not encompass all possible variations in signal states, a controlled level of noise is introduced to improve generalization. Based on prior-art research on data augmentation for training enhancement, the model applies white noise compensation at 5% rate.
- Implementation of constraint mechanisms: To prevent unrealistic state transitions, the model enforces a set of constraints representing the necessary and sufficient conditions for signal state changes. For instance, parameters such as center frequency, channel bandwidth, power level, and signal-to-noise ratio are constrained within predefined ranges derived from domain-specific knowledge and empirical data.
- Design of the reward function: The reward function is designed to evaluate the accuracy of predicted signal parameters relative to ground truth data. Specifically, it non-linearly incorporates three key metrics: deviations in predicted center frequency, bandwidth, and noise duration. This function operates at each time step and reflects both the state characteristics and temporal prediction errors, thereby incentivizing accurate, real-time signal state estimation.

### B. Algorithm Design

This section presents the detailed design of the training algorithm, which is developed based on the DDPG algorithm. The proposed algorithm comprises the following key components:

- RL agent is responsible for controlling and executing the training process.
- Primary and Target Actor Networks: Two deep neural networks, the Primary Actor Network and Target Actor Network, are employed to directly output signal parameters predicted from the current signal state. These networks are associated with weight parameters denoted as $\theta_\mu$ and $\theta_{\mu'}$, respectively.
- Primary and Target Critic Networks: Two additional deep neural networks, the Primary Critic Network and Target Critic Network, are used to evaluate the Q-value associated with each state-action (i.e., signal state–predicted signal parameters) pair. Their corresponding weights are represented by $\theta_Q$ and $\theta_{Q'}$, respectively.
- Experience Replay Buffer: A memory that stores training data in the form of past experiences, enabling the RL agent to sample mini-batches of data for training and thereby improving stability and efficiency.

The architecture of the Primary Actor Network and Primary Critic Network are illustrated in Figure 1. These networks are enhanced with residual connections and spectral regularization techniques to improve training stability during the weight update process. Residual connections help mitigate issues related to vanishing gradients in deep networks, while spectral regularization constrains the spectral norm of the network layers to promote robustness and generalization.

The operational flow of the proposed algorithm is illustrated in Figure 2, which comprises two main processes:

- Data collection process: This stage is responsible for generating the dataset required for training. It involves repeated interactions between the RL agent and the signal

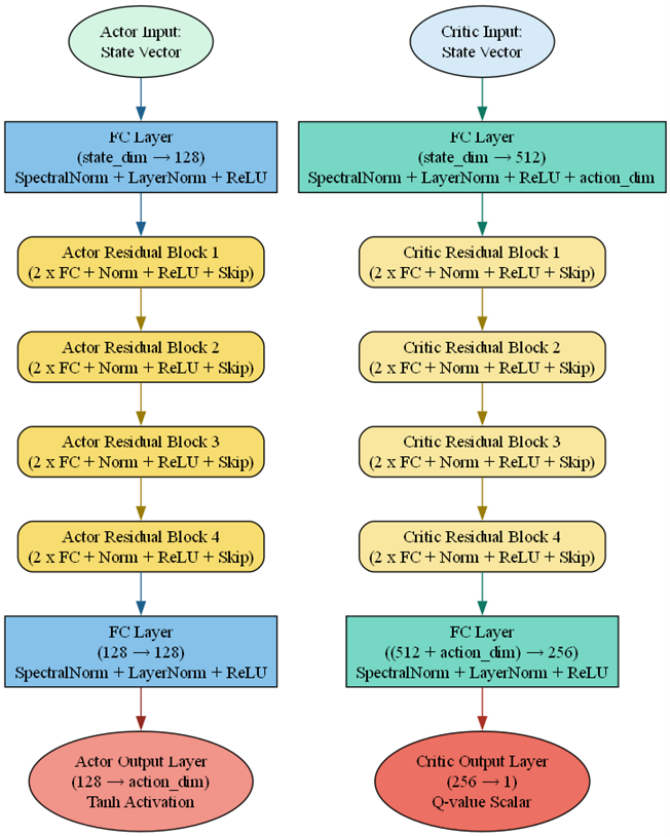


Fig. 1. The architecture of actor and critic neural networks.

dynamics environment. At each time step $t$, the RL agent observes a current environmental state, denoted as $S(t)$, which captures key characteristics of the signal environment. Based on this state, the RL agent utilizes current weights of the actor network to process a set of signal parameters, encapsulated in an action vector $a(t)$. This vector typically includes the predicted central frequency, channel bandwidth, and signaling time for the next time step. Following this, the RL agent computes the reward $r(t)$ by comparing the predicted parameters with the actual values. Concurrently, the new environmental state $S(t+1)$ is observed using actual collected signal data. Each interaction results in a tuple $<S(t), a(t), r(t), S(t+1)>$, which is stored in the Experience Replay Buffer. This data process iteratively saves experience data into the memory buffer. Since it has limited capacity, the oldest data entries are replaced when the buffer reaches its maximum size. It is important to note that this data collection phase operates offline, utilizing pre-collected data from the entire BDWS dataset to construct a comprehensive experience buffer.

- Training process for frequency hopping prediction: In this stage, the RL agent is trained to optimize its signal prediction capability. The training of the neural networks is initiated once a sufficient number of training samples have been accumulated in the Experience Replay Buffer.
  - Actor network training: The training begins by sampling the state $S(t)$ from the dataset. The Primary Actor Network uses this state input to output the predicted action $a(t)$. This output is then passed to the Primary Critic Network, which evaluates it and computes a corresponding Q-value. A policy loss function, denoted as $L(\theta_\mu)$), is formulated to quantify the discrepancy between expected and actual Q-value performance. The Adam optimizer is employed to update the weights of the Primary Actor Network accordingly. The weights of the Target Actor Network is copied from the Primary one after each $G$ steps.
  - Critic network training: Training of the Critic Network begins by sampling both state-action pairs $<S(t), a(t)>$. The Primary Critic Network computes the Q-value $Q(S(t), a(t); \theta_Q)$ for each sample. In parallel, the next state $S(t+1)$ is processed through the Target Actor and Target Critic Networks to produce a predicted Q-value $Q(S(t+1), a(t+1); \theta_{Q'})$. These values are used to calculate the Bellman loss function $L(\theta_Q)$, which is minimized using the Adam optimizer to update the Critic Network's weights. The weights of the Target Critic Network is copied from the Primary one after each $G$ steps.

## IV. Evaluation

### A. Parameter Settings

The signal environment is simulated using data collected from the frequency hopping signals of a drone DJI Mavic 3 Pro. The drone operates in a pre-configured frequency hopping mode based on the Ocusync 3 protocol, alternating frequency in the 2.4GHz bands. During data acquisition, the drone is positioned at a fixed distance of 50 meters from the receiver, with the receiving antenna oriented directly toward the drone to optimize signal capture. The collected data consists of estimated BDWS values, which are processed by an FPGA module from the raw I/Q signal data. The specific configuration parameters used for the training algorithm are summarized as: number of episodes is 30000, batch size is 32, discounting factor is 0.9, and learning rate is 0.001.

Reward function is designed as follows. The reward function is designed to quantify the accuracy of the predicted signal parameters, including the central frequency $fc$, channel bandwidth $bw$, and signal transmission time. Specifically, a frequency range is defined by combining the central frequency and bandwidth values. The predicted frequency range and signaling time are then compared to the corresponding actual values to compute both an accuracy metric and a penalty for deviation. The accuracy metric, denoted as $A$, reflects the proportion of overlapped area between the predicted and actual frequency ranges. It is a normalized value within the interval [0,1], where $A = 1$ indicates perfect alignment between the predicted and actual ranges, and $A = 0$ indicates no overlap. Similarly, the penalty metric, denoted as $B$, measures the cost associated with prediction error. It also ranges from

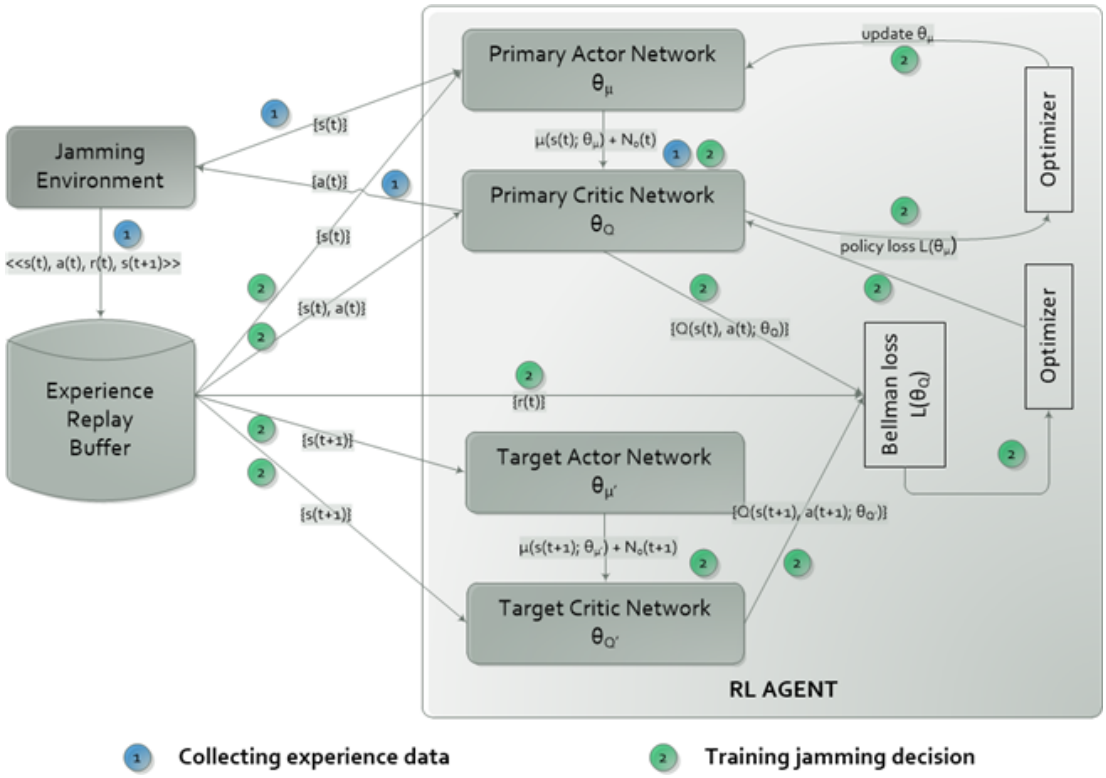


Fig. 2. Deep Reinforcement Learning for predictive modeling of frequency hopping signals.

[0,1], where $B = 0$ corresponds to perfect prediction (or no prediction made), and $B = 1$ reflects complete mismatch. Let $M$ and $N$ be the respective weights assigned to the accuracy and penalty terms. The intermediate reward function, denoted as $f_r$, is defined as: $f_r = M \cdot A - N \cdot B$. The maximum value of $f_r$ is $M$, achieved when $A = 1$ and $B = 0$, i.e., the predicted parameters perfectly match the actual values. Conversely, the minimum value is $-N$, which occurs when $A = 0$ and $B = 1$, indicating a complete mismatch. Separate rewards are computed for the frequency range prediction, $f_r(F)$ and the signaling time prediction, $f_r(T)$. These are combined into the total reward function $R$, formulated as: $R = a \cdot f_r(F) + b \cdot f_r(T)$, where $a$ and $b$ are the weights assigned to the frequency and time prediction sub-rewards, respectively, and satisfy the constraint $a + b = 1$. As a result, the total reward value $R$ lies within the interval $-[N, M]$. Training objective is to maximize the value of the total reward function $R$. In the initial configuration, the parameters are set as follows: $M = 0.9$, $N = 0.1$, $a = 0.7$, and $b = 0.3$.

### B. Numerical Results

This sub-section presents numerical results for evaluating the performance of the proposed framework. The extent to which the reward function converges during the training process is analyzed, as it reflects the model's ability to learn optimal actions over time. The reward evolves to converge after approximately 5000 episodes. At the convergence, the reward is achieved at almost 97.4% of its maximum theoretical value, indicating the model has achieved a reasonably high level of predictive accuracy.

## V. Conclusion

This study presented a novel approach with DRL to accurately predict the UAV frequency hopping signals for efficient UAV jamming. Specifically, the proposed framework is developed based on an enhanced DDPG algorithm, maximizing a reward function that reflects the accuracy of predicting the frequency-hopping parameters, i.e., central frequency, transmission channel bandwidth, and signal transmission time. A frequency hopping model is constructed using pre-collected signal data that practically verify the feasibility, enabling greater flexibility in model customization and facilitating the evaluation of various training configurations. The convergence behavior observed during training indicates that the proposed framework is capable of predicting multiple signal parameters with reasonable accuracy. The final reward function value achieved during training reached almost 97.4% of its theoretical maximum value, representing the optimal case in which predicted parameters perfectly match the actual signal environment.

## References

[1] D. S. Lakew, A.-T. Tran, N.-N. Dao, and S. Cho, "Intelligent Self-Optimization for Task Offloading in LEO-MEC-Assisted Energy-Harvesting-UAV Systems," IEEE Transactions on Network Science and Engineering, vol. 11, no. 6, pp. 5135–5148, Nov.-Dec. 2024.

[2] T. H. Moges, T. P. Truong, D. S. Lakew, T. H. Huong, V. T. Hoang, N.-N. Dao, and S. Cho, "Age of information-aware trajectory optimization for time-sensitive UAV systems in uplink SCMA networks," to appear in Elsevier Computer Networks, 2025.
[3] V. D. Tuong, W. Noh, and S. Cho, "Spatial Deep Learning-Based Dynamic TDD Control for UAV-Assisted 6G Hotspot Networks," IEEE Transactions on Industrial Informatics, vol. 20, no. 9, pp. 11092–11102, Sept. 2024.
[4] S. Shrestha, M. Ababneh, S. Misra, H. M. Cathey Jr, R. Vishwanathan, M. Jansen, J. Choi, R. Bobba, and Y. Jang, "A Comprehensive Survey of Unmanned Aerial Systems' Risks and Mitigation Strategies," 2025, arXiv:2506.10327. [Online]. Available: https://arxiv.org/abs/2506.10327
[5] S. Liu, "Wi-Fi Energy Detection Testbed (12MTC)," 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC
[6] K. Eves and J. Valasek, "Adaptive control for singularly perturbed systems examples," Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree
[7] P. Thiele, L. Bernadó, D. Löschenbrand, B. Rainer, C. Sulzbachner, M. Leitner, and T. Zemen, "Machine Learning Based Prediction of Frequency Hopping Spread Spectrum Signals," in 2023 Proc. IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1–6, Sept. 2023.
[8] M. T. Khan, A. Z. Sha'ameri, and M. Mun'im Ahmad Zabidi, "Classification of FHSS signals in a multi-signal environment by artificial neural network," International Journal of Computing and Digital Systems, 11(1), 2022.