

EEE 493/494 Industrial Design Project Committee 4 Report

Deep Learning-Assisted Time–Frequency Domain Drone Detection

Derin Öğrenme Tabanlı Zaman–Frekans Düzleminde İHA Tespiti



İhsan Doğramacı Bilkent University

Department of Electrical and Electronics Engineering

Members of Group: B5

Sıla Bulduk	22103508	Tuna Şahin	22201730
Çağan Herek	22202929	Mısra Simge Uysal	22201527
Özgü Melek Özyörük	22202590	Çınar Kanbur	22202213

Company Advisors: Rıdvan Soysal, Yakup Ali İşcen

Academic Advisor: Asst. Prof. Melih Baştopçu

Course Coordinator: Mehmet Alper Kutay

Project TA: Batuhan Uykulu

Course TAs: Mustafa İlyas Çalışkan, Elif Ahsen Çakmakçı

Date: 13.05.2026

Company Information:

Meteksan Savunma is a leading high-technology company within Turkey's defense industry, developing advanced radar, electro-optic, sonar, communication, and simulation systems, focusing on meeting the operational needs of the Turkish Armed Forces with innovative, domestically developed solutions that strengthen Turkey's defense capabilities.

Contents

List of Figures	III
List of Tables	V
1 Project Summary	1
2 Motivation and Novelty	1
2.1 Project Motivation	1
2.2 Similar Solutions	2
2.2.1 STFT-Based Drone Detection from RF Signals	2
2.2.2 FFT-Based Drone Detection from RF Signals	2
2.3 Similar Products	2
2.3.1 Aartos X2	2
2.3.2 AirWarden Essentials	3
2.4 Project Novelty	3
2.5 Possibility of a Publication	3
2.6 Possibility of a Patent	4
2.7 End-User of the Project	4
3 Requirements	4
3.1 Functional Requirements	4
3.1.1 Requirements Concerning Input Data Format	5
3.1.2 Requirements Concerning Detection Algorithms	5
3.1.3 Requirements Concerning Classification Algorithms	6
3.1.4 Requirements Concerning the GUI	6
3.1.5 Optional Requirements	6
3.2 Non-Functional Requirements	7
4 System Architecture	8
5 Methods and Implementation Details	9
5.1 Work Breakdown Structure and Project Plan	9
5.1.1 Work Package 1	10
5.1.2 Work Package 2	11
5.1.3 Work Package 3	11
5.1.4 Gantt Chart and Milestones	11
5.2 Tools, Methods and Progress	14
5.2.1 Tools	14
5.2.2 Methods	15

5.2.3	Progress	25
5.2.4	Encountered Risks	43
6	Results, Discussions, and Future Directions	43
6.1	Results	43
6.1.1	Timing Related Results	43
6.1.2	Detection Accuracy Results	44
6.1.3	Classification Related Results	44
6.1.4	Signal Parameter Extraction Results	49
6.2	Discussions and Lessons Learned	49
6.3	Future Directions	50
7	Detailed Equipment List	50
	References	51
	Appendices	54
1	Detailed Gantt Chart	54
2	Non-Functional Requirements, Standards	55
3	Detailed System Architecture	56
4	Reassigned STFT Generation and VGG19 Model	57
5	FFT-Based Custom Segmentation Network - SegNet	58
6	FFT-Based Network Accuracy Results	59
7	Confusion Matrices of Trained Classification Models	61
8	FFT-Based GPU Training Results	66
9	Ablation Study Results	68
10	CRNN Preprocessing Pipeline	69
11	Drone ID Telemetry GNU Radio Block Diagram	70
12	Graphical User Interface Pictures	71
13	Detailed Timing Analysis of the GUI	75
14	Terminology Relating to the Project	76

List of Figures

1	Big Picture	9
2	Work Breakdown Structure (WBS)	10
3	Project Milestones	14
4	Detection and Differentiation of Interfering Signals	18
5	Stage 3 Example Input and Output	19
6	Original and Isolated Emitter Spectrograms	20
7	Multiple Detection Algorithm Outputs	20
8	Multiple Classification Algorithm Example	21
9	FFT-Pipeline Process Flowchart	26
10	FFT-Pipeline Process Flowchart	26
11	Comparison of spectrograms generated from the same RF signal sample using 2048-point and 1024-point STFTs. The main time-frequency characteristics are preserved in the 1024-point representation, while the computational cost is reduced.	29
12	Representative STFT spectrogram examples of the three additional UAV-related classes included in the extended 9-class dataset.	33
13	Drone ID Telemetry Results	36
14	Annotated GUI of the Application	36
15	STFT and Spectrogram Display Panels	37
16	Detailed system architecture diagram highlighting the hardware and software components and their interconnections.	56
17	Example Reassigned Spectrogram (DJI MAVIC AIR 2)	57
18	Confusion Matrix for VGG19 Model on Reassigned Spectrogram Dataset . .	57
19	SegNet Inputs, Labels, and Outputs	58
20	Dataset Augmented FFT-Based CNN Confusion Matrix	59
21	Bin-Wise Normalised Confusion Matrix for SegNet	59
22	Classification-Wise Normalised Confusion Matrix for SegNet	60
23	Confusion matrix of the ResNet18 model for the 6-class dataset.	61
24	Confusion matrix of the ResNet34 model for the 6-class dataset.	62
25	Confusion matrix of the ResNet50 model for the 6-class dataset.	62
26	Confusion matrix of the ResNet101 model for the 6-class dataset.	63
27	Confusion matrix of the SwinV2-Tiny Transformer model for the 6-class dataset.	63
28	Confusion matrix of the ViT-B/16 model for the 6-class dataset.	64

29	Confusion matrix of the ViT-L/16 model for the 6-class dataset.	64
30	Confusion matrix of the MobileNetV3-Small model for the 6-class dataset. .	65
31	Confusion matrix of the ResNet18 model for the extended 9-class dataset. .	65
32	Gaussian Model Confusion Matrices	66
33	Gaussian Model Loss	66
34	Gaussian-Free Model Confusion Matrices	67
35	Gaussian-Free Model Loss	67
36	Block Diagram of the FFT-CRNN Preprocessing Pipeline	69
37	Block Diagram of the STFT-CRNN Preprocessing Pipeline	69
38	Conceptual architecture of the CRNN classification model	69
39	GNU Radio Block Diagram for Drone ID, Part 1/2	70
40	GNU Radio Block Diagram for Drone ID, Part 2/2	70
41	GUI, Theme: Default Dart	71
42	GUI, Theme: Creme	71
43	GUI, Theme: Night Ops	72
44	GUI, Theme: Resin	72
45	Detailed Current GUI Screenshot (Theme = Default Light)	73
46	GUI at the Time of CM3	74

List of Tables

1	Project Timeline of Work Package 1	12
2	Project Timeline of Work Package 2	12
3	Project Timeline of Work Package 3	13
4	Possible Interference Cases	17
5	Model size comparison of the trained 6-class architectures.	31
6	Overview of All Obtained Recordings	35
7	End-to-End Average Latency of the Main Processing Workers	44
8	Error Rate for Detection Systems	44
9	Classification Performance Metrics Obtained for the FFT-Based Model . . .	44
10	Offline classification results of the trained 6-class models using the same 1024-point STFT test dataset.	45
11	Offline classification result of the extended 9-class ResNet18 model.	46
12	GUI-integrated field-test performance of the trained models.	47
13	Error Rate for Detection Systems	49
14	Detailed Equipment List (*Currency Conversion According to 8 May. 2026 Rates)	50
15	Detailed Gantt Chart	54
16	Comparison of CNN model architectures, accuracy results, file sizes, FFT confusion matrix times, bias observations, and datasets.	60
17	Performance comparison of the FFT-based CNN trained on the GPU-accelerated Colab pipeline with and without the Gaussian noise augmentation layer. Both models share identical preprocessing, mixed-precision training, warmup + cosine annealing schedule, label smoothing, and test sets.	67
18	Ablation Study Results of Different ResNet18-Based Architectures	68
19	RecorderWorker Average Latency Breakdown	75
20	DetectorWorker Average Latency Breakdown	75
21	MultipleDetectionWorker Average Latency Breakdown	75
22	Latency Breakdown of Average Other Detection Related Processes	76

1 Project Summary

The increasing use of unmanned aerial vehicles (UAVs) has introduced significant challenges related to spectrum congestion, interference, and security risks. Since commercial drones primarily operate in the 2.4 GHz and 5.8 GHz ISM bands, reliable RF-based detection and identification within these crowded frequency ranges have become increasingly important. This project, conducted in collaboration with Meteksan, aimed to develop a real-time RF signal analysis framework for drone detection, identification, and model classification. The finalized system addressed three main tasks: drone versus non-drone detection, separation of drone signals from common wireless communication signals, and classification of drone models based on their RF signatures. In the initial phase, publicly available datasets, namely DroneRF and RFUAV, were used for algorithm development and benchmarking. Raw IQ samples were processed using FFT-based spectral analysis and STFT-based time–frequency representations. Both classical signal processing techniques and data-driven approaches were employed. FFT-based feature extraction and threshold-based detection methods provide lightweight processing, while convolutional neural networks (CNNs) are used for robust classification. Development and simulations were carried out using SDR hardware, together with GNU Radio, and Python-based deep learning frameworks. The test and validation procedure included offline simulations on labeled datasets, controlled interference experiments within the 2.4 GHz and 5.8 GHz bands, and real RF signal collection using USRP B200mini. System performance was evaluated using quantitative metrics such as detection accuracy, classification accuracy, false alarm rate, and processing latency. At the end of the project, a development-ready, real-time drone detection and identification framework supported by a graphical user interface (GUI) has been successfully delivered, providing intuitive visualization of detected drone activity and signal characteristics. The finalized project has achieved a detection accuracy of 99.78% and a classification accuracy of 100% (6-class ResNet18). The end-to-end average latency of classification and detection have been measured as 1818.8 ms and 127.43 ms respectively. Overall, the system provides an effective real-time solution for the detection, classification, monitoring, and analysis of drone-related RF activity in complex wireless environments.

2 Motivation and Novelty

2.1 Project Motivation

The proliferation of UAVs has led to an increase in their usage for malign purposes [1–3]. As such, in many contexts it is important to take measures against potentially malignant applications of UAV technology. Identification is the first major step in such a procedure. The need for a system that is able to effectively detect, classify and identify drones in the vicinity is the main motivating factor for this project. This project is to become property of

Meteksan, which may use it in internal applications that cannot be disclosed due to privacy reasons or improve it further by integrating it with their own systems.

2.2 Similar Solutions

Several existing studies have addressed RF signal detection and classification using different signal representations and learning-based techniques.

2.2.1 STFT-Based Drone Detection from RF Signals

In recent studies, a focus on RF-based drone detection was observed, including the transformation of raw IQ samples into time–frequency representations using the short-time Fourier transform (STFT) [4–6]. Within the approaches, STFT spectrograms were highlighted for their ability to capture different time-varying spectral patterns of UAV transmissions, which is crucial in learning-based detection or statistical processing methods. Although works including the RFUAV study [7] capture effective drone detection and identification under different signal conditions using STFT-derived spectrograms, their sole reliance on time–frequency information lacks complementary FFT-based spectral feature inclusion.

2.2.2 FFT-Based Drone Detection from RF Signals

The methods proposed by recent studies were seen to capture RF-based drone detection with the aid of spectral feature extraction through Fourier transform operations, indicating dependence on frequency-domain analysis alone [8, 9]. These methods include IQ signal segmentation and FFT-based frequency domain transformation to obtain power spectral representations, characterizing the signal energy distribution. In Amasri’s work, Welch’s power spectral density estimation, based on FFT processing of windowed signals, was implemented to derive deep learning imposed compact spectral features for drone detection and identification [10]. Although the results indicate FFT-derived spectral signatures’ effectiveness in detecting UAV presence, their insufficiency in capturing time–frequency variations addressed by STFT-based approaches were perceptible.

2.3 Similar Products

Some similar products are given in this section as examples to provide points for comparison with our project.

2.3.1 Aartos X2

Aaronia’s Aartos X2 is a ”low-cost and mobile drone detection system with a long range of up to 5 km, capable of decoding drone logs and pinpointing their location” [11]. It is a portable device that makes use of the RF signals that drones emit for detection, weighing approximately 0.5 kg. It uses the RTSA software, developed by the same company, that

has various advanced functionalities including jammer setups, multi-frequency 3D direction finding, 3D spectrum analysis, deep learning target classification among others [12]. It is made in Germany.

2.3.2 AirWarden Essentials

Airwarden Essentials is a subscription-based drone detection system that relies on the Federal Aviation Association (FAA) requirement of drone remote ID broadcasting to deliver real-time insights into the detected drones [13]. It includes a receiver that needs to be installed for the system to work, which is a detriment regarding the portability aspect. It costs \$415 per month at minimum, which in comparison to our solution is less cost-effective in the long run.

2.4 Project Novelty

The completed project presents a drone detection and identification approach that combines FFT-based spectral analysis with STFT-based time–frequency representations. While existing RF-based solutions generally rely on a single representation domain, this project exploited the complementary strengths of both approaches. FFT-based features provide stable spectral characteristics, whereas STFT-based spectrograms capture time-varying signal behavior that is particularly useful for learning-based detection. The system further integrates classical signal processing methods to classify and identify multiple drones at once. By employing these approaches together, the system aimed to achieve improved robustness against co-band interference such as Wi-Fi and Bluetooth signals, while remaining suitable for real-time implementation. This combined use of classical signal processing and deep learning-oriented representations constitutes a distinctive contribution to RF-based drone detection methodologies.

2.5 Possibility of a Publication

The system developed in this project provided a practical framework for RF-based drone detection, identification, and classification operating in crowded ISM bands. The project combined classical signal processing techniques with deep learning–based classification methods in order to achieve quick and accurate performance when analyzing UAV-related RF signals under realistic interference conditions. This comprehensive approach enabled the evaluation of different feature extraction methods and classification strategies within a unified real-time framework. In addition, a detailed ablation study was conducted for the ResNet18-based model to investigate the contribution of different architectural modifications to the overall system performance and computational efficiency.

The results and analysis obtained throughout the process form a sufficient basis for a journal publication or a technical conference such as the IEEE International Conference on Acoustics,

Speech, and Signal Processing (ICASSP), and the International Conference on Unmanned Aircraft Systems (ICUAS). In particular, comparative studies on FFT and STFT-based representations, robustness against non-drone wireless signals, and trade-offs between detection accuracy and processing latency are of potential academic interest. Although the current findings demonstrate highly promising performance, additional large-scale evaluations, hyperparameter analysis, broader environmental testing, and more extensive comparative analyses would likely be beneficial for strengthening the contribution toward publication-level research.

2.6 Possibility of a Patent

The probability of this project receiving a patent is relatively low due to the existence of similar products and solutions. There is a considerable amount of literature that examines applications of deep learning in the context of drone detection, classification and identification [14–16]. [17] is a patented product that uses AI to detect and classify drones from multiple RF receivers. In the patent [18] classification takes place using the micro-doppler effects of UAVs and neural networks. The novelty of the project comes essentially from the way the methods are employed rather than the originality of the methods and as such does not constitute a sufficient basis for the acquisition of a patent.

2.7 End-User of the Project

The end-users of the project are essentially technical personnel of corporations or institutions related to the military and the defense industry, who have technical experience and knowledge regarding the basics of drone communications and spectral analysis, and value the analytic capabilities and the performance of the system more compared to the visual interface. In addition to institutional deployment, the proposed system may also be utilized by armed forces personnel in operational field environments for the detection and classification of nearby drones, and identification of potentially foreign or unauthorized UAV platforms. The system may be deployed by them as an auxiliary module for defensive purposes.

3 Requirements

This section presents the functional and non-functional requirements of the system in accordance with the project scope and design objectives. The requirements describe the capabilities, constraints, and operational conditions of the final product as precisely as possible.

3.1 Functional Requirements

Functional requirements cover both user-dependent and autonomous capabilities of the system. User interaction consists of monitoring and visualization of system outputs through

the GUI and changing system parameters in real-time. All core signal processing tasks, including drone signal detection, RF parameter estimation, and drone type classification, are performed autonomously by the system in real time.

3.1.1 Requirements Concerning Input Data Format

- **IFSR-01:** The system shall read raw data from files using standard complex baseband IQ data formats compatible with SDR systems (e.g., float32 or int16).
- **IFSR-02:** The system shall read raw or spectrum data from real-time SDR streams provided by the SDR and USRP B200mini using GNU Radio and vendor-supported SDR drivers (e.g., UHD : USRP Hardware Driver).

These requirements define the supported input data formats and interfaces of the system. Standard SDR-compatible IQ data formats are used to ensure interoperability with commonly used SDR software frameworks, while real-time communication with SDR hardware is achieved through existing vendor-supported drivers without implementing custom communication protocols. As of CM4 all of the requirements concerning input data format are satisfied.

3.1.2 Requirements Concerning Detection Algorithms

- **DSR-01:** The system shall detect drone signals by analyzing frequency-domain representations of the received RF signals.
- **DSR-02:** The system shall correctly estimate the bandwidth of detected drone signals with an acceptable error range of ± 0.5 MHz.
- **DSR-03:** The system shall correctly estimate the center frequency of detected drone signals with an acceptable error range of ± 0.5 MHz.
- **DSR-04:** The detection time shall be less than one second after drone signals become present in the environment.
- **DSR-05:** The detection algorithms shall be implemented in real time to the GUI application.
- **DSR-06:** The system shall detect signals with a minimum accuracy of 95% when the SNR is greater than or equal to 15 dB.
- **DSR-07:** The system shall differentiate drone signals from interfering sources such as Wi-Fi, Bluetooth, and other co-band emitters.

These requirements define the real-time detection behavior and robustness of the system. As of CM4 all of the requirements concerning detection algorithms are satisfied.

3.1.3 Requirements Concerning Classification Algorithms

- **CLSR-01:** The classification algorithms shall be implemented in real time to the GUI application.
- **CLSR-02:** The classification time shall be less than two seconds.
- **CLSR-03:** The system shall classify each drone type, including DJI Mavic Air, DJI Mini 3, and DJI Mini 4 Pro.
- **CLSR-04:** The classification accuracy shall be higher than 95%.
- **CLSR-05:** The classification accuracy shall be higher than 90% under Wi-Fi and Bluetooth interference.
- **CLSR-06:** The system shall be capable of performing STFT-based classification.

These requirements define the functionality and performance of the classification stage. They ensure accurate and timely identification of drone types using different signal representations and under both single- and multi-drone scenarios. As of CM4 all of the requirements concerning classification algorithms are satisfied.

3.1.4 Requirements Concerning the GUI

- **GUIR-01:** The GUI shall display the real-time spectrum graph of the receiver.
- **GUIR-02:** The GUI shall display the real-time spectrogram of the receiver.
- **GUIR-03:** The GUI shall display an emitter list containing detected signals.
- **GUIR-04:** The GUI shall display the bandwidth of the detected drone signal.
- **GUIR-05:** The GUI shall display the center frequency of the detected drone signal.
- **GUIR-06:** The GUI shall display the estimated SNR of detected signals.
- **GUIR-07:** The GUI shall be able to support real-time configuration of the SDR's center frequency.

The GUI requirements ensure that the system outputs are clearly visualized for monitoring and analysis. Real-time graphical feedback is provided to support interpretation of detection and classification results. As of CM4 all of the requirements concerning the GUI are satisfied.

3.1.5 Optional Requirements

- **OPTSR-01:** The system shall detect drones employing a frequency hopping algorithm.
- **OPTSR-02:** The system shall identify Wi-Fi and Bluetooth signals.
- **OPTSR-03:** The GUI shall visualize detected signals on the spectrum graph.
- **OPTSR-04:** The system shall analyze DroneID data for supported drones.

- **OPTSR-05:** The system shall support adding new drone types to the detection and classification databases.
- **OPTSR-06:** The system shall be able to detect up to 3 drones at once.

Optional requirements have been considered after all mandatory functional requirements were successfully fulfilled. The optimal requirements were all satisfied with the exception of OPTSR-01.

3.2 Non-Functional Requirements

- **NFSR-01:** Privacy: Captured RF signals shall be used only for detection and classification; no content extraction, storage, or external sharing shall be performed.

This requirement is important for privacy and protection concerns. It ensures that the gathered data is not abused for other purposes.

- **NFSR-02:** Maintainability: The codebase shall be clean and well-documented with a clear structure to support modification and troubleshooting.

This requirement serves as insurance for the maintainability and accessibility of the project by keeping the code clean.

- **NFSR-03:** Power Efficiency: CPU and GPU utilization shall be optimized for minimal power consumption and stable real-time performance. The maximum allowable power consumption shall be 20 W.

This power constraint refers to the additional computational and processing overhead introduced by the deployed system components, rather than the baseline power consumption of the host computer.

- **NFSR-04:** Environmental Operating Conditions: The system shall be operable in both indoor and outdoor environments without requiring special design measures.

Environmental constraints are limited to normal operating conditions, as the system is designed as a software-driven solution deployed on commercial SDR platforms.

- **NFSR-05:** Weight: The complete hardware kit (SDR, antennas, and mounts) shall be portable by one person and weigh no more than 5 kg.
- **NFSR-06:** Size: The maximum allowable size of the hardware setup shall be $1200 \times 600 \times 300$ mm.
- **NFSR-07:** Minimum Computer Requirement: The system shall require a 3 Gbps Ethernet port for SDR connectivity, a high clock speed CPU (i5 or equivalent), and an NVIDIA GPU with at least 6 GB VRAM.
- **NFSR-08:** Cost: The maximum allowable system cost shall be \$11,543.

This cost constraint corresponds to the total cost of the SDR devices and drones provided by the company. Since all required hardware resources are supplied by the

company, no additional budget limitation is imposed on software development activities.

- **NFSR-09:** Safety: The system shall comply with IEC 61010-1 to ensure safe operation of electrical and laboratory equipment.
- **NFSR-10:** Health: RF exposure levels shall comply with ICNIRP 2020 and IEEE C95.1 standards for human electromagnetic field exposure.
- **NFSR-11:** Cultural Factors: No cultural constraints are applicable to the system.
- **NFSR-12:** Complied Standards: The system shall adhere to relevant standards including IEC 61010-1, ETSI EN 301 489 (EMC), IEEE C95.1, and RoHS 2011/65/EU. In addition, the system shall be compatible with common communication and data transfer standards used by SDR platforms, including IEEE 802.3 (Ethernet), USB 2.0/3.0, IEEE 802.11 (Wi-Fi), and Bluetooth. The explanations of these standards are available in Appendix 2

The non-functional requirements define system constraints related to performance, safety, usability, and compliance. The implemented system complies with all of the non-functional requirements.

4 System Architecture

The system architecture of the finalized system is shown in Fig. 1. The hardware layers are given in dark red boxes, the software layers are given in green boxes, and the power connections are indicated with gold-colored lines. At the hardware level, RF signals transmitted by drones are captured using a software-defined radio (SDR) platform, USRP B200mini. It operates as a compact, bus-powered SDR that communicates with the host computer through a USB 3.0 interface, which provides both data transfer and 5 V power. In this configuration, RF signals are delivered to the SDR through standard coaxial (SMA) RF connections.

On the software side, the Ettus Research USRP Hardware Driver (UHD) provides low-level control and configuration of the SDR hardware. GNU Radio is used on the host computer for real-time signal acquisition and IQ sample streaming. GNU Radio is interfaced with the Python-based processing environment through the *radioconda* framework, which enables integrated SDR library management, and real-time IQ data streaming. It also makes it possible to have a seamless interaction between signal acquisition, preprocessing pipelines, and deep learning inference modules within the unified software environment of Python.

A self-collected dataset captured by USRP B200mini is used for the training of FFT-based detection and STFT-based classification models. On the other hand, the data acquired by the SDR in real-time is processed by the finalized FFT and STFT-based preprocessing pipelines in Python to generate spectrum and spectrogram data to be processed by the DL models.

MATLAB was additionally utilized during the initial development stages for FFT/STFT preprocessing optimization and timing analysis. Finally, the resulting outputs the of de-tection and classification models and signal parameter extraction methods are presented to the user through a PyQt-based graphical user interface, enabling real-time monitoring and analysis of drone activity.

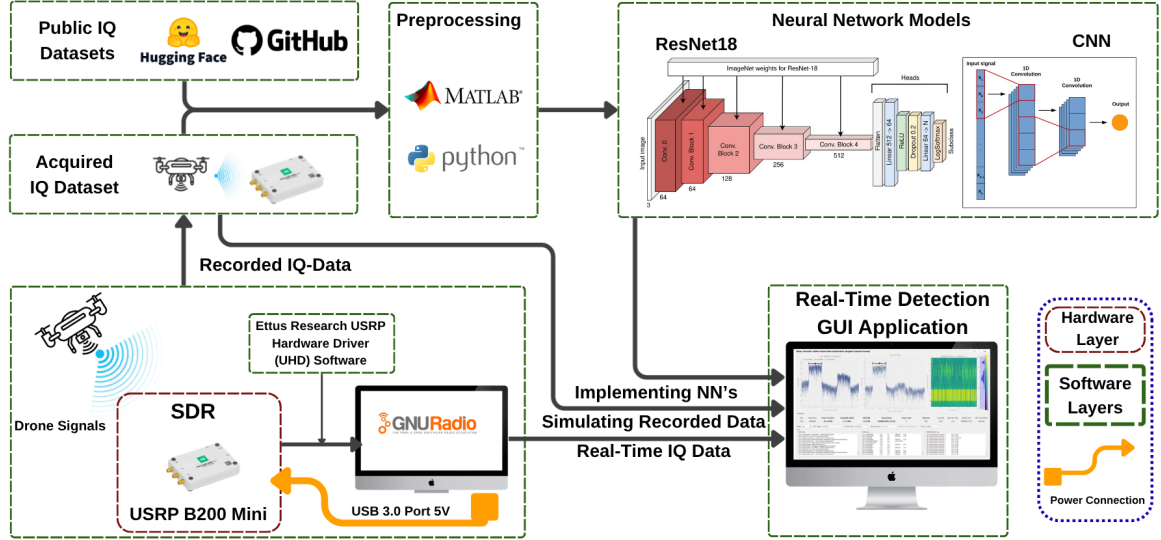


Figure 1: Big Picture

5 Methods and Implementation Details

In this section, the work breakdown structure, tools, methods and project progress are presented.

5.1 Work Breakdown Structure and Project Plan

This project is divided into three work packages: Signal Processing and Neural Network Training (given in pink), Acquisition of Drone Signals via SDR (given in purple), and Real Time Implementation and GUI Development (given in blue). The work breakdown structure is presented in Fig 2. The filled boxes indicate the main tasks in the respective work packages. The sub-tasks branch out from the main tasks they are related to in order. Dashed boxes represent ongoing tasks, dotted boxes represent completed tasks, and filled boxes denote the primary task modules. As of CM4, all tasks have been completed.

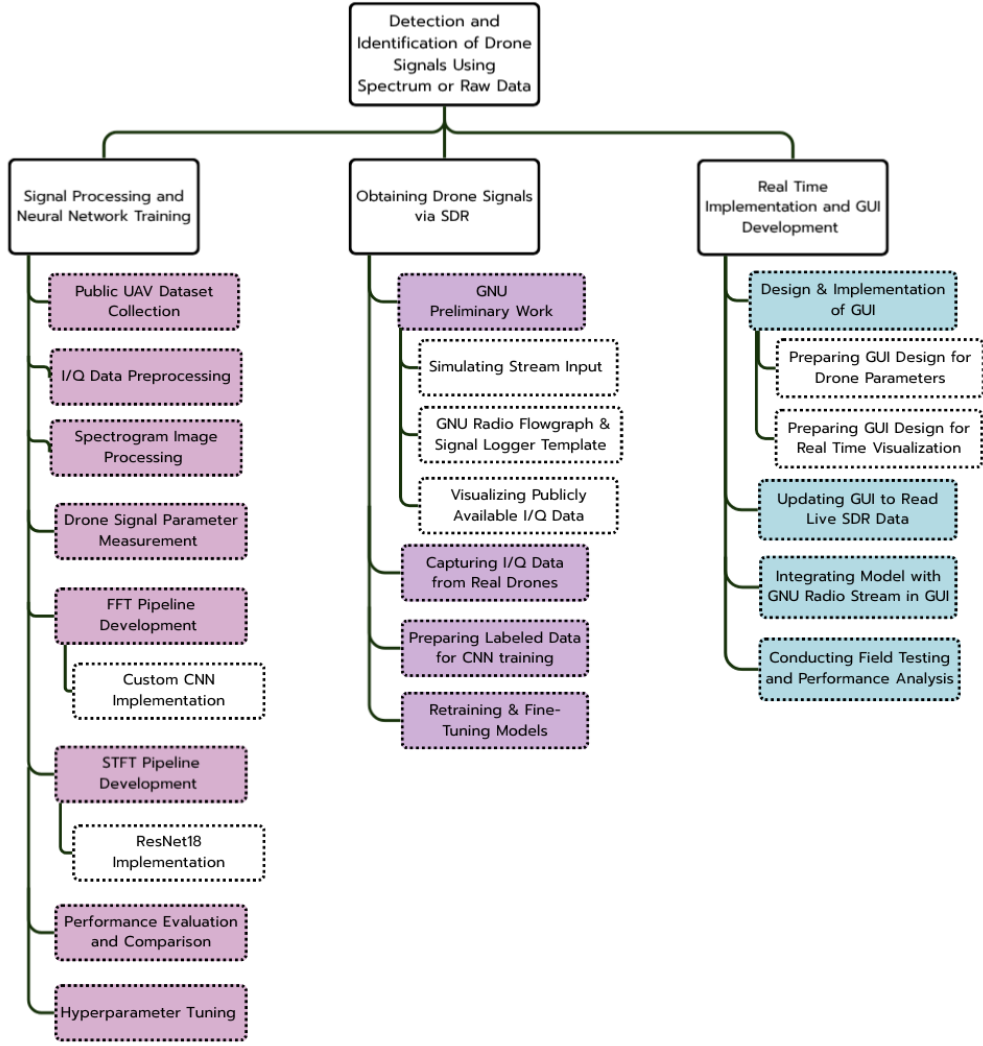


Figure 2: Work Breakdown Structure (WBS)

5.1.1 Work Package 1

The first work package focused on the development of signal processing techniques and deep learning models for drone signal detection and identification. This work package included the collection of publicly available UAV datasets, followed by the preprocessing of IQ data obtained from DroneRF and RFUAV sources [19, 20]. Spectrogram representations were generated after optimization of STFT parameters. In parallel, algorithms for the signal characteristics extraction such as SNR, bandwidth, and center frequency were developed and evaluated. Then, FFT and STFT-based processing and learning approaches were implemented to support detection and classification tasks. Several deep learning architectures that support FFT and STFT input, including custom and ResNet-based convolutional neural networks, were trained and evaluated. Model performance was further improved through hyperparameter optimization and comparative benchmarking.

5.1.2 Work Package 2

The second work package addressed the acquisition of drone signals using USRP B200 mini and the preparation of real-world data for model training and validation. Initial experiments featured GNU Radio signal flowgraphs and data logging mechanisms [21]. Simulated data streams from publicly available RFUAV IQ recordings were employed to verify the processing pipeline [20]. After the recordings were visualized and analyzed, additional IQ data was captured from real drones operating in environments with Wi-Fi and Bluetooth interference. The training dataset was expanded using the collected data. The collected data was labeled and organized for training. To enhance the model’s ability of generalising to real captured signals, transfer learning was utilized. With the constant expansion of the training dataset with self-recorded data the DL models were further fine-tuned with improved robustness.

5.1.3 Work Package 3

The third work package focused on real-time system implementation and system-level integration. Within this scope, the trained detection and classification models, along with the signal processing algorithms were integrated into a live SDR-based signal processing chain. A graphical user interface was developed to support real-time visualization and monitoring of drone signal parameters, as well as interaction with the underlying processing modules. The system was configured to operate on live SDR input streams, enabling real-time processing. Further field testing and performance analysis have been conducted to evaluate the operational stability, responsiveness, and overall effectiveness of the integrated system under realistic conditions.

5.1.4 Gantt Chart and Milestones

The project schedule is summarized in the Gantt chart presented in Tables 1, 2, 3, which indicate task durations, assigned team members, and milestone timing. A more detailed and horizontally oriented version of the Gantt chart is provided in Appendix 1. The green cells represent work completed until now, blue cells represent upcoming tasks, and yellow cells indicate the end of the timeline.

	Task	Task Owner
1	WP1	Leader: Melek, Tuna
1.1	Public UAV Dataset Collection	Melek, Sila, Tuna
1.2	I/Q Data Preprocessing (DroneRF and RFUAV datasets)	Sila, Tuna
1.3	Spectrogram Image Processing via Resizing, Color Mapping (RFUAV)	Melek, Sila
1.4	Drone Signal Parameter Measurement (SNR, BW, f _c)	Tuna
1.5	FFT Pipeline Development in Python for Detection/Classification	Tuna
1.6	STFT Pipeline Development in Python/MATLAB for Detection/Classification	Melek, Sila
1.7	Custom CNN Implementation for FFT Input in Python	Tuna, Misra
1.8	ResNet Implementation for Spectrogram Input in Python	Melek, Sila
1.9	Performance Evaluation by Comparing Against the Detection and Classification Benchmarks	Melek, Misra, Sila, Tuna
1.10	Hyperparameter Tuning for Accuracy and Real Time Deployment	Melek, Sila, Tuna
1.11	Processing of Acquired Data Through Developed Pipelines	Melek, Misra, Sila, Tuna
1.12	Retraining (Transfer Learning Strategy) and Fine-Tuning Models with Real Captures	Çınar, Çağan, Melek, Tuna
1.13	Training of Alternative Models for Performance Comparison and Exploration of Alternative Solutions	Melek, Sila, Tuna
1.14	Development of Multiple Detection and Classification Algorithms	Melek, Misra, Sila, Tuna
1.15	CRNN Pipeline Development and Training in Python	Misra

	October	November	December	January	February	March	April	May	Jun
Task Owner	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2
Leader: Melek, Tuna									
Melek, Sila, Tuna									
Sila, Tuna									
Melek, Sila									
Tuna									
Tuna									
Melek, Sila									
Tuna, Misra									
Melek, Sila									
Melek, Misra, Sila, Tuna									
Melek, Sila, Tuna									
Melek, Misra, Sila, Tuna									
Çınar, Çağan, Melek, Tuna									
Melek, Sila, Tuna									
Melek, Misra, Sila, Tuna									
Misra									

Table 1: Project Timeline of Work Package 1

	Task	Task Owner
2	WP2	Leader: Çağan
2.1	Preliminary Work	Çınar, Çağan
2.1	GNU Radio Flowgraph & Signal Logger Template Development	Çınar, Çağan
2.3	Simulating Stream Input for GNU Radio Testing	Çınar, Çağan
2.4	Capturing IQ Data From Real Drones in Required Environments Such as Under Wi-Fi/Bluetooth Interference	Çınar, Çağan, Melek
2.5	Development of Drone ID Acquisition Algorithms	Çınar, Çağan

	October	November	December	January	February	March	April	May	Jun
Task Owner	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2
Leader: Çağan									
Çınar, Çağan									
Çınar, Çağan									
Çınar, Çağan									
Çınar, Çağan, Melek									
Çınar, Çağan									

Table 2: Project Timeline of Work Package 2

	Task	Task Owner
3	WP3	Leader: Tuna
3.1	Beginning GUI Development for Drone Signal Parameters	Tuna, Sila
3.2	Beginning GUI Development for Real-Time Signal Visualization	Çağan, Tuna, Sila
3.3	Updating GUI to Read Livestream SDR Input	Çağan, Tuna, Sila
3.4	Enabling User Control of SDR and Signal Processing Parameters in the GUI	Tuna, Sila
3.5	Displaying Live Emitter Information in the GUI	Tuna, Sila
3.6	Integrating Signal Processing Algorithms in the GUI	Tuna
3.7	Integrating Models with Real-Time Datastream in the GUI	All
3.8	Conducting Field Testing and Performance Analysis	Tuna, Sila
3.9	Implementing DronelD Acquisition System to the GUI	Tuna, Çınar, Çağan

	October	November	December	January	February	March	April	May	Jun
Task Owner	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4	W1 W2 W3 W4
Leader: Tuna									
Tuna, Sila									
Çağan, Tuna, Sila									
Çağan, Tuna, Sila									
Tuna, Sila									
Tuna, Sila									
Tuna									
All									
Tuna, Sila									
Tuna, Çınar, Çağan									

Table 3: Project Timeline of Work Package 3

The project milestones were defined in accordance with the main development phases and were explicitly linked to the corresponding work packages. The first milestone was associated with WP1 and targeted reliable detection and classification of drone signals using signal processing and deep learning techniques. The finalized project reached this milestone with a detection accuracy of 99.78% and a classification accuracy of 100% (6-class ResNet18). The second milestone corresponded to WP2 and focused on obtaining drone signals via SDR with the support of proper GNU Radio configuration. This milestone was achieved as the finalized end-to-end system is capable of acquiring real time IQ data from SDRs. The final milestone was linked to WP3 and represented successful real-time system integration and live operation of the end-to-end system. This milestone was achieved through the development of a complete end-to-end real-time RF-based drone detection and classification framework capable of continuous SDR-based signal acquisition, preprocessing, deep learning inference, and live detection/classification under realistic operating conditions.

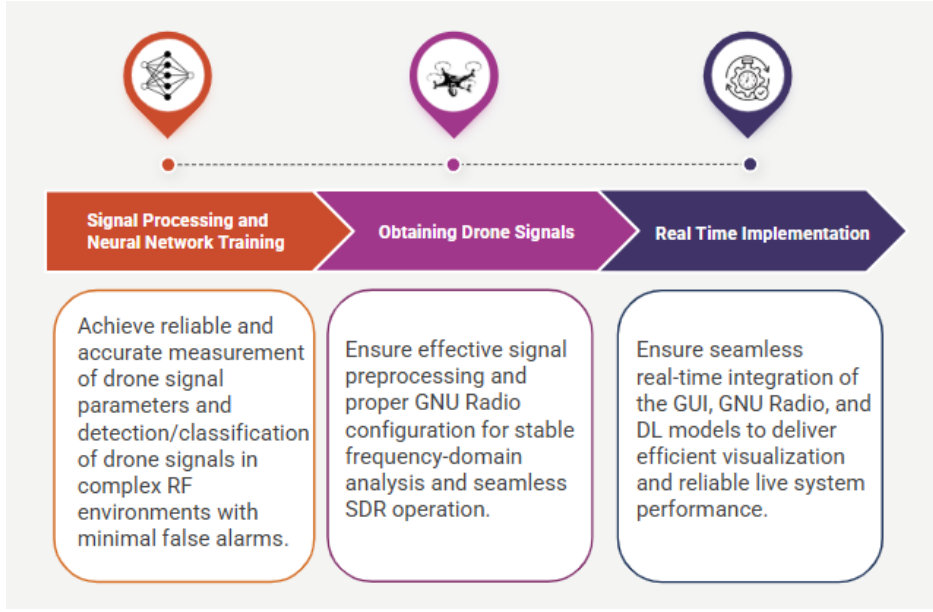


Figure 3: Project Milestones

5.2 Tools, Methods and Progress

In this section, the tools and methods used in the project are introduced according to the work packages they were applied in and the progress in the course of the entire project is discussed.

5.2.1 Tools

Throughout the development process for various parts of the system, different tools have been used. These tools consist of different software for development and various hardware components to implement real-life operation. The tools are categorized according to their prevalence in each work package.

5.2.1.1 Tools Used in WP1

5.2.1.1.1 Publicly Available UAV Datasets

The RFUAV [20] dataset that features raw data and spectrograms is used to train deep learning models. The raw data obtained from RFUAV is also used to assess the performance of the STFT based pipeline across a range of operational parameters. Moreover, the robustness and generalization capabilities of our deep learning models are validated by using raw IQ samples from the RFUAV dataset.

5.2.1.1.2 Python

Python is used for all data preprocessing and deep learning applications. Libraries such as Tensorflow [22] and Pytorch [23] enable developers to write simple code that implements deep learning architectures. Algorithms for SNR and bandwidth estimation are coded in Python.

5.2.1.1.3 MATLAB

MATLAB has been utilized to support the refinement of the preprocessing pipelines through FFT/STFT parameter optimization and detailed timing analysis. [24].

5.2.1.2 Tools Used in WP2 and WP3

5.2.1.2.1 Software Defined Radios (SDRs)

The USRP B200mini has been provided by Meteksan. This SDR has served as means of gathering raw data from the environment in the 2.4 GHz and 5.8 GHz bands, where most drone communication signals are transmitted. The SDR was configured with the sampling rate 40 MS/s.

5.2.1.2.2 GNU Radio Companion

GNU Radio Companion is the main environment used for interfacing the SDR with the computer. It is used in applications concerning the acquisition and processing of raw data. It facilitates the simple and efficient implementation of signal processing algorithms through its built-in compiled blocks. Its "file source" feature can be used to simulate data gathered from datasets. It also has a GUI capability, QT widgets can be implemented as blocks in a GNU Radio Companion flowchart.

5.2.1.2.3 Drone Models

The drone models DJI Mavic Air, DJI Mini 3 and DJI Mini 4 Pro have been provided by Meteksan for testing purposes and data collection.

5.2.1.3 Tools Used in WP3

5.2.1.3.1 PyQt

The PyQt framework is used for the design of interactive GUI that can communicate with the signal processing pipeline using parallel threads [25].

5.2.2 Methods

Different algorithms were developed during the development process and were consequently deployed in the final system. The methods and their mathematical foundations have been explained accordingly in their sections.

5.2.2.1 Methods Used in WP1

5.2.2.1.1 Temporal Consistency Filtering

Let $N_d(T)$ denote the number of drone-positive detections observed inside a sliding time window of duration T . The system declares a certain drone detection when

$$\frac{N_d(T)}{T} \geq R_{th}, \quad (1)$$

where R_{th} is the minimum required detection rate. In the implemented system, this logic is applied using a recent detection timestamp buffer. Each possible drone detection appends its timestamp to the buffer, and old timestamps outside the window are removed. If the remaining timestamp count corresponds to a sufficiently high detection rate, the GUI displays the drone state as certain.

5.2.2.1.2 Drone Signal Parameter Measurement

The bandwidth and center frequency of signals present in the spectrum are estimated using a threshold-based detection approach. The detection of different signals in the band is done in three stages using different thresholding methods.

Stage 1: Signal Detection and Location

Let $X[k]$ denote the log-magnitude of the discrete Fourier transform (DFT) of the received signal. First, the signal's envelope is extracted in order to have consistent threshold crossings this envelope is calculated as:

$$X_e[k] = \max(X[k - i]), \quad \forall i \in ([-50, 50] \cup \mathbb{Z}) \quad (2)$$

Using this envelope, the noise floor is estimated from the spectrum and denoted as P_{noise} . A detection threshold is then defined by adding an adaptive margin Δ above this noise floor, chosen according to the expected signal-to-noise ratio (SNR), as

$$T_1 = P_{noise} + \Delta, \quad \Delta = \min(5, SNR_{peak} \times 0.4) \quad (3)$$

The magnitude spectrum is compared against this threshold to produce a binary decision vector $b[k]$, defined as

$$b[k] = \begin{cases} 1, & X_e[k] \geq T, \\ 0, & X_e[k] < T. \end{cases} \quad (4)$$

Consecutive sequences of ones in the binary vector $b[k]$ correspond to individual signal components in the frequency domain.

Stage 2: Interference Detection and Seperation

In order to separate instances where other signals might be interfering with drone signals, we need to check deviations in signal intensity in each band. The binary decision vector $b[k]$ is split into individual vectors $b_i[k]$ for each consecutive sequence of ones. Then another threshold T_2 is applied to each vector independently.

$$T_2 = \max(5, P_{noise} + SNR_{peak} \times 0.7) \quad (5)$$

This second threshold allows for the differentiation of closeby signals that have different SNRs. Then, the decision vector obtained for each $b_i[k]$ using this threshold is defined as:

$$b_i^{(2)}[k] = \begin{cases} 1, & X_e[k]b_i[k] \geq T_2, \\ 0, & X_e[k]b_i[k] < T_2. \end{cases} \quad (6)$$

This $b_i^{(2)}[k]$ vector is the decision vector for the higher SNR signal. Then in order to extract the signals independently, we firstly identify the edges of these decision vectors:

$$b_i[k] = 1, \forall k \in [k_{i,start}, k_{i,end}], \quad b_i[k] = 0, \forall k \notin [k_{i,start}, k_{i,end}] \quad (7)$$

Similarly, we define the same variables for $b_i^2[k]$

$$b_i^{(2)}[k] = 1, \forall k \in [k_{i,2,start}, k_{i,2,end}], \quad b_i^{(2)}[k] = 0, \forall k \notin [k_{i,2,start}, k_{i,2,end}] \quad (8)$$

Using these edge values, we check five possible cases for each identified $b_i[k]$:

Case	Conditions for each case
There is only one signal	$k_{i,start} - k_{i,2,start} < \mathcal{M}, \quad k_{i,end} - k_{i,2,end} < \mathcal{M}$
The high SNR signal is to the left	$k_{i,start} - k_{i,2,start} < \mathcal{M}, \quad k_{i,end} - k_{i,2,end} > \mathcal{M}$
The high SNR signal is to the right	$k_{i,start} - k_{i,2,start} > \mathcal{M}, \quad k_{i,end} - k_{i,2,end} < \mathcal{M}$
The high SNR signal is in the middle	$k_{i,start} - k_{i,2,start} > \mathcal{M}, \quad k_{i,end} - k_{i,2,end} > \mathcal{M}$
The signal has other signals on both sides	$k_{i,start} - k_{i,2,start} > \mathcal{M}, \quad k_{i,end} - k_{i,2,end} > \mathcal{M}$

Table 4: Possible Interference Cases

The cases 4 and 5 are indistinguishable and significant information loss would have occurred in both cases hence we discard those signals. In cases 2,3 the decision vector $b_i^{(2)}[k]$ is split into two other vectors. For case two, the vector is split at $k_{i,2,end}$. For case three, the vector is split at $k_{i,2,start}$. For case one no changes are necessary to the vector. After the necessary splits, the vectors obtained in stage 2 are added to the list obtained in stage 1, along with the other $b_i[k]$'s. An example output for case 3 is shown in Fig. 4:

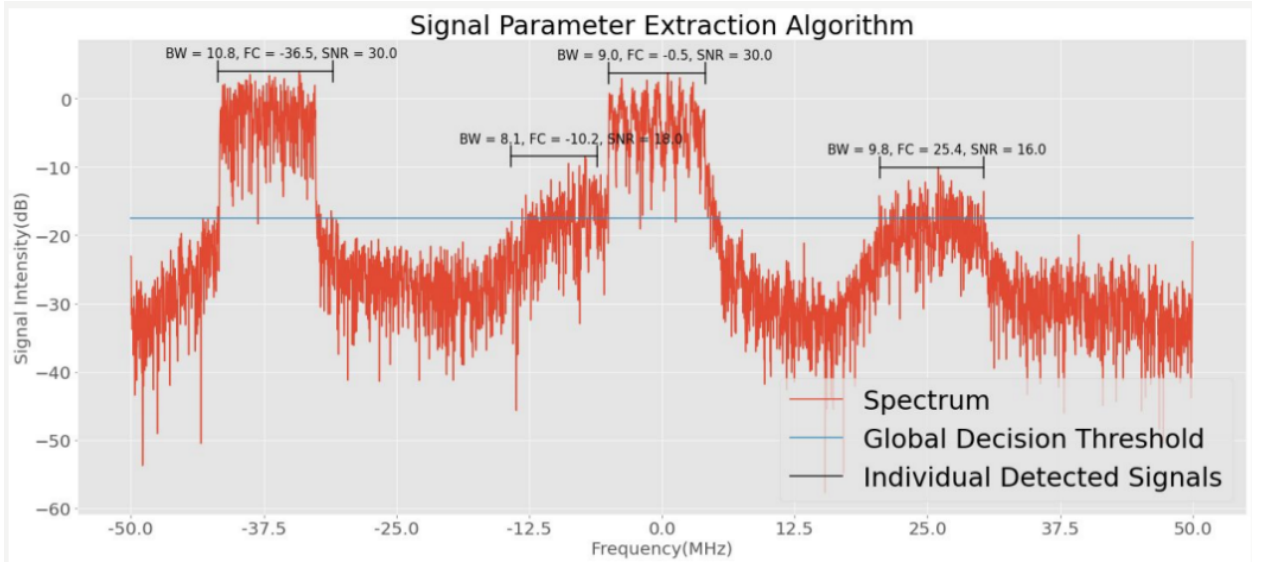


Figure 4: Detection and Differentiation of Interfering Signals

Stage 3: Signal Grouping and Separation Correction

For signals like Wi-Fi with non-uniform amplitudes, signals with SNR close to T_1 can register as multiple threshold crossings. To correct these phenomena, a third threshold T_3 is introduced. This new threshold is again administrated for each $b_i[k]$ separately.

$$T_3 = \max(5, T_1/2) \quad (9)$$

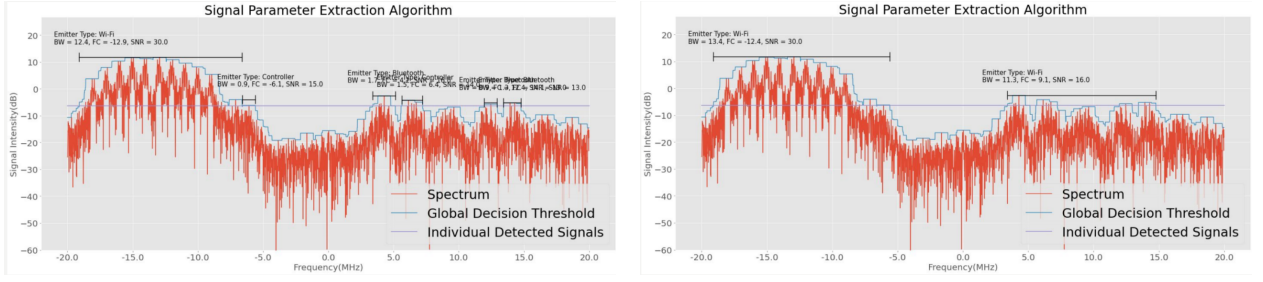
The comparison decision vector against T_3 for each $b_i[k]$ is calculated as:

$$b^{(3)}[k] = \begin{cases} 1, & X_e[k] \geq T_3, \\ 0, & X_e[k] < T_3. \end{cases} \quad (10)$$

It is important to note that this comparison is applied without the mask of $b_i[k]$ as was the case in stage 2. Then $b^{(3)}[k]$ is again separated into different sequences as $b_j^{(3)}[k]$. Then, the $b_j^{(3)}[k]$ vector that encompasses the current $b_i[k]$ is found as:

$$J = \underset{j}{\operatorname{argmax}} \langle b_j^{(3)}[k], b_i[k] \rangle \quad (11)$$

The crossings that belong to the same emitter but fall under different $b_i[k]$'s register together under $b_i^{(3)}[k]$ comparison vector. Therefore, if the J value is the same for multiple $b_i[k]$, they are removed from the list of $b_i[k]$'s and instead replaced with $b_j^{(3)}[k]$. An example for this scenario is given in Fig. 5



Before Stage 3

After Stage 3

Figure 5: Stage 3 Example Input and Output

After these 3 steps, each signal's corresponding decision vector is obtained as $b_i[k]$. The center frequency, bandwidth and SNR information of each signal is then obtained. Again, for the i -th detected signal, let the corresponding sequence span the index interval $k \in [k_{i,start}, k_{i,end}]$. The bandwidth of the detected signal is obtained by counting the number of consecutive bins above the threshold.

$$B_i^{(\text{bins})} = k_{i,end} - k_{i,start} + 1. \quad (12)$$

The frequency is converted to normalised frequency by dividing by the total number of samples, and then converted to actual frequency by multiplying with the sampling frequency.

$$B_i = B_i^{(\text{norm})} f_s. \Rightarrow B_i^{(\text{norm})} = \frac{B_i^{(\text{bins})}}{N}, \quad (13)$$

The center frequency of the i -th detected signal is determined as the midpoint of the corresponding consecutive-ones segment, which is converted to physical frequency as:

$$k_{i,center} = \frac{k_{i,start} + k_{i,end}}{2}, \Rightarrow f_{i,center} = \frac{k_{i,center}}{N} f_s. \quad (14)$$

This frequency is then added to the baseband frequency to obtain the absolute center frequency. The SNR and peak signal intensity is obtained simply from $b_i[k]$ as:

$$P_i = \max(X[k]b_i[k]), \quad SNR = P_i - P_{noise} \quad (15)$$

By identifying and processing all such segments independently, the algorithm is capable of detecting multiple signals and extracting their parameters from same spectrum.

5.2.2.1.3 Multiple Detection and Classification Algorithms

Using the signal parameter extraction algorithm's ability to detect multiple signals' parameters, this algorithm is developed to classify each detected signal. The $b_i[k]$ vectors are used as masks to isolate each emitter in the spectrum, leading to multiple spectrums $X_i[k]$ for each emitter. In order to simulate realistic spectrums, the bins outside of the mask should be replaced with noise instead of setting them to zero. A noise vector $n[k]$ is generated as zero-mean white gaussian noise with the necessary standard deviation to match the actual

environment. The updated spectrums are of the form:

$$X_i[k] = b_i[k]X[k] + n[k](1 - b_i[k]) \quad (16)$$

An example of the isolated spectrums are shown in Fig. 6

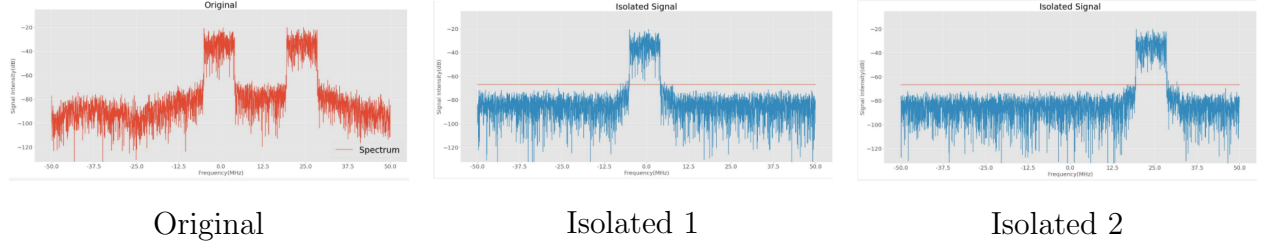


Figure 6: Original and Isolated Emitter Spectrograms

These isolated spectrums are then fed to the necessary neural network to classify each spectrogram and obtain its corresponding emitter type. Along with the binary decision vectors $b_i[k]$, this algorithm can obtain the SNR, center frequency, bandwidth and emitter type of each signal in a given spectrum. An example result, obtained using this algorithm is shown if Fig. 7

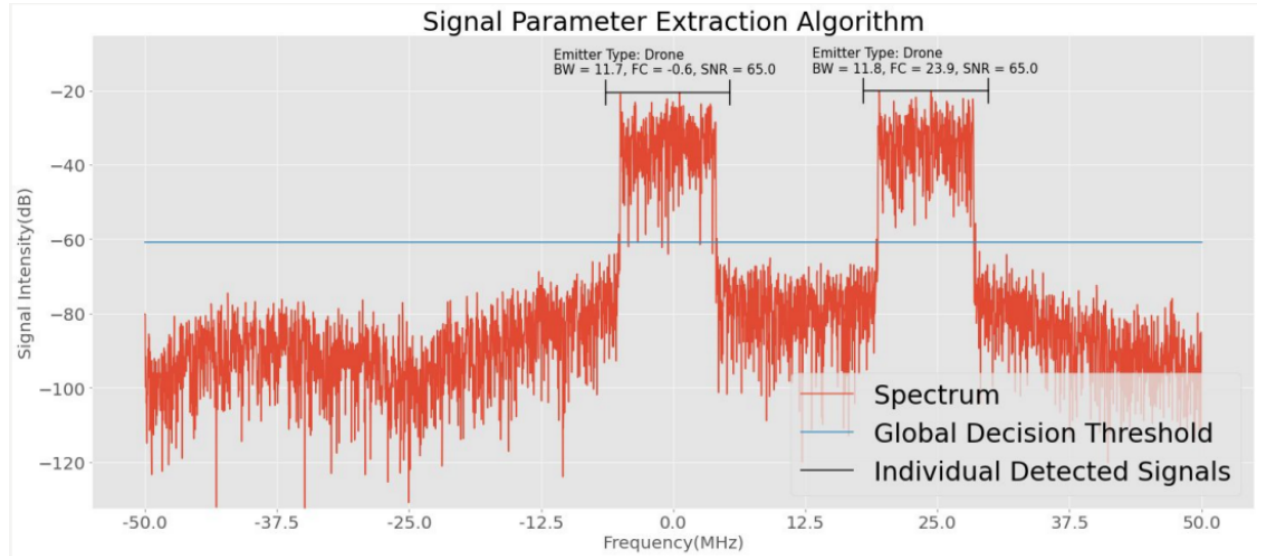


Figure 7: Multiple Detection Algorithm Outputs

The same algorithm is then scaled to 2 dimensions to be able to perform multiple classifications from the same spectrogram $X[t, f]$. For each drone signal, other drone signals in the spectrogram are suppressed and the drone signal is shifted to the center of the frequency axis to match the training data of the trained models. The mask for each drone is given as:

$$m_i[t, f] = \text{rect} \left(\frac{f - f_{c,i}}{BW_i} \right), \forall t \quad (17)$$

Hence for the i 'th drone signal, the isolated spectrogram is calculated by setting every other drone's mask to artificial noise while keeping the signal $X[t, f]$ for samples outside the mask

for other drones. The final isolated spectrogram is given as:

$$X_i[t, f] = X[t, f] \times \left(1 - \sum_{j \neq i} m_j[t, f]\right) + n[t, f] \times \left(\sum_{j \neq i} m_j[t, f]\right) \quad (18)$$

Finally, to shift each spectrogram to the center, we shift each spectrogram by the corresponding drone signals center frequency:

$$X'_i[t, f] = X_i[t, f - f_{c,i}] \quad (19)$$

An example process of the algorithm is shown in Fig 8.

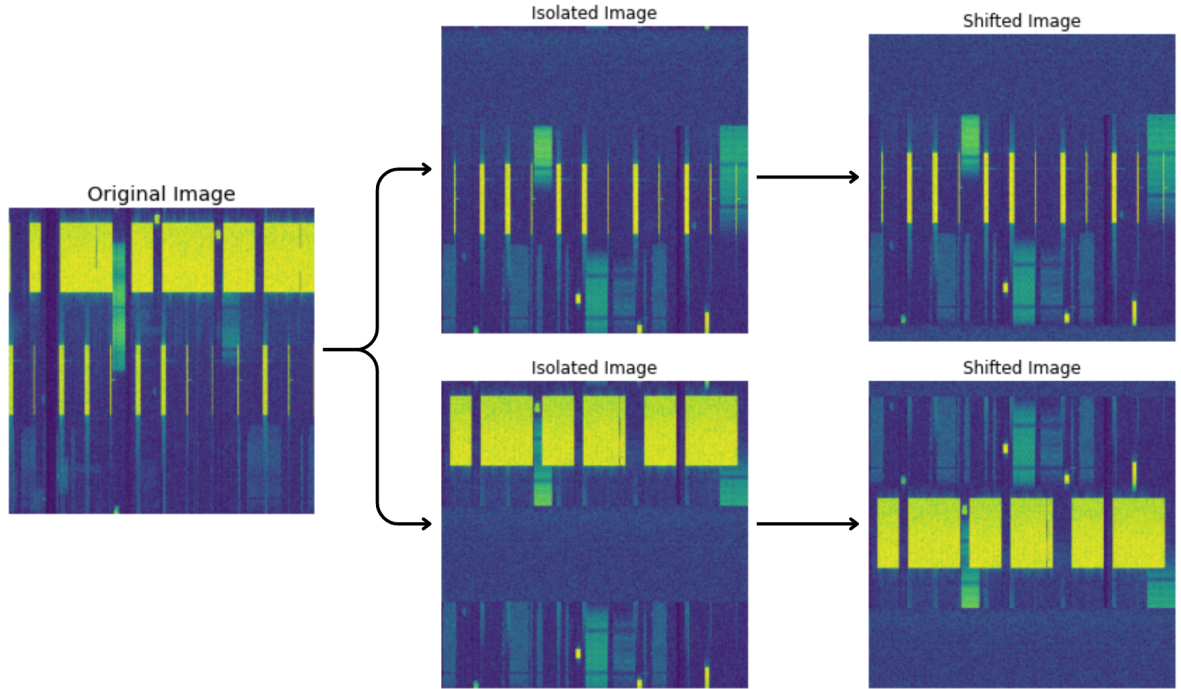


Figure 8: Multiple Classification Algorithm Example

5.2.2.1.4 Neural Network Models

Custom CNN Architecture

The developed model is a 1D CNN designed to process FFT-based spectral data. The current architecture accepts an input vector of 4096 neurons that represent the discretized frequency domain. Feature extraction is performed through a series of convolutional layers with ReLU activation and 1D pooling. A Global Average Pooling layer interfaces the convolutional blocks with a classification head composed of dense layers utilizing L2 regularization. The network concludes with a 3-neuron output layer with Softmax activation to classify the detected RF signals into the following categories: Noise, Drone/Controller(Remote), and Wi-Fi/Bluetooth.

FFT-Based Custom Segmentation Network (SegNet)

Another FFT-based model is proposed to classify each signal and extract their bandwidth and center frequency information. The architecture has 4096 input neurons that again represent the FFT. This model features 19 Conv1D layers, 3 max-pooling layers, 3 upsampling layers, 3 skip connections, one dropout layer, and a softmax output that classifies each FFT bin into one of 3 classes. The overall output of the network is $[4096, 3]$ where the model outputs the probability of each bin separately. The progress related to this model is given in Appendix 5.

ResNet18

The ResNet18 architecture is implemented for the drone classification task. The model accepts 640×640 spectrograms. The design utilizes stacked residual blocks with identity skip connections which allows the model to learn residual mappings, improving training stability. The output of this model is a probability distribution across drone models included in the training set.

5.2.2.1.5 SNR Estimation Algorithm

The signal-to-noise ratio (SNR) is estimated directly from the received complex baseband samples using a power spectral density (PSD)-based approach. Let $r[n]$ denote the discrete-time received signal. The algorithm begins by computing the PSD of $r[n]$ via Welch's method, which improves robustness by averaging periodograms obtained from overlapping windowed segments. The resulting PSD is represented in the logarithmic domain as

$$P(f) = 10 \log_{10}(\text{PSD}\{r[n]\}), \quad (20)$$

where $P(f)$ denotes the spectral power distribution across frequency bins. To identify noise-dominated regions of the spectrum, a percentile-based selection is employed. Low-energy frequency components are assumed to correspond primarily to noise, and the noise floor is estimated as

$$P_{\text{noise}} = \text{percentile}(P(f), \alpha), \quad (21)$$

with $\alpha = 20\%$. High-energy spectral components are associated with signal presence. The set of signal-dominated frequency bins is defined as

$$\mathcal{F}_s = \{f \mid P(f) \geq \text{percentile}(P(f), \beta)\}, \quad (22)$$

where $\beta = 90\%$. The average signal power is then computed by

$$P_{\text{signal}} = \frac{1}{|\mathcal{F}_s|} \sum_{f \in \mathcal{F}_s} P(f). \quad (23)$$

The SNR estimate is obtained as the difference between the average signal power and the estimated noise floor,

$$\text{SNR}_{\text{PSD}} = P_{\text{signal}} - P_{\text{noise}} \text{ (dB)}. \quad (24)$$

The primary purpose of this algorithm is to provide a direct quantitative estimate of the SNR of the received waveform. Although SNR values are also used as an intermediate quantity in bandwidth estimation procedures, those approaches treat SNR as a supporting metric rather than a final output. In contrast, the method described here is explicitly designed as a standalone SNR estimation algorithm.

5.2.2.1.6 Downsampling from 100 MHz to 40 MHz

The RFUAV dataset consists of complex baseband IQ samples acquired at a sampling rate of $f_s^{(\text{orig})} = 100$ MHz. However, the data collected using the USRP B200mini is constrained by a maximum achievable sampling rate of $f_s^{(\text{target})} = 40$ MHz. The mismatch between the sampling rates of the RFUAV dataset and the data we collected, causes inconsistency across training data. In order to ensure consistency between datasets and enable meaningful comparative analysis, it is necessary to downsample the RFUAV data to match the lower sampling rate.

The sampling rate conversion ratio is given as

$$\frac{f_s^{(\text{target})}}{f_s^{(\text{orig})}} = \frac{40}{100} = \frac{2}{5} = \frac{L}{M},$$

where $L = 2$ is the upsampling factor and $M = 5$ is the downsampling factor. The resampling procedure is conducted through the upsampling, filtering, and downsampling steps. First, the signal $x[n]$ is upsampled by inserting $L - 1$ zeros between samples:

$$x_u[n] = \begin{cases} x[n/L], & n \equiv 0 \pmod{L} \\ 0, & \text{otherwise} \end{cases}.$$

Next, an anti-aliasing low-pass filter $h[n]$ needs to be applied. The frequency response of the anti-aliasing filter must satisfy

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| \leq \frac{\pi}{\max(L, M)} \\ 0, & \text{otherwise} \end{cases}$$

to prevent aliasing and spectral distortion.

$$x_f[n] = (x_u * h)[n] = \sum_{k=-\infty}^{\infty} x_u[k] h[n - k].$$

Finally, the filtered signal is downsampled by a factor M :

$$y[n] = x_f[nM].$$

This resampling ensures that both datasets share the same sampling rate and effective bandwidth, preserving the validity of subsequent signal processing and training steps.

5.2.2.1.7 The Method of Reassignment

The method of reassignment is a method that can be applied to overcome the intrinsic flaws of time-frequency representations such as the tradeoff between temporal and spectral resolution and biased estimation of signal instantaneous frequency and group delay [26,27]. It essentially makes spectrogram images more readable by mapping the magnitudes to modified time-frequency pairs. The relation between the reassigned and original time-frequency pairs is given by the equations below as indicated in [28].

$$\hat{t}_x(t, w) = t - Re \left\{ \frac{F_x(t, w; T_z(t)) F_x^*(t, w; z)}{S_x(t, w)} \right\} \quad (25)$$

$$\hat{w}_x(t, w) = w + Im \left\{ \frac{F_x(t, w; D_z(t)) F_x^*(t, w; z)}{S_x(t, w)} \right\} \quad (26)$$

Where the STFT using the window z is denoted as $F_x(t, w; z)$ and $T_z(t) = tz(t)$, $D_z(t) = \frac{dz(t)}{dt}$, $S_x(t, w) = |F_x(t, w; z)|^2$. More information about the implementation of and progress in this method along with examples of reassigned-STFTs are available in Appendix 4.

5.2.2.2 Methods Used in WP3

5.2.2.2.1 Multithreading

To ensure real time operation visualition and data acquisition in a single-thread GUI, computationally expensive tasks such as continuous data acquisition, STFTs, and continuous visualization can cause the main loop to freeze or lag.us data acquisition, STFTs, and continuous visualization can cause the main loop to freeze or lag. To avoid this issue, the individual tasks are seperated into their independent worker threads while the main thread is reserved for the user interaction and visualisation. Interthread communications are handled using mechanisms such as signals, slots, and buffers which protect asynchronous access. Since the main thread is reserved for visualisation and user access, the information is sent to the main thread which is then displayed to the user in the main thread instead of letting individual threads access the visual interface. This multithreaded architecture improves real time performance, preserves interactibility, and it allows for a modular system where we can disable and enable individual components without affecting the main system.

5.2.3 Progress

5.2.3.1 Progress up to CM1

The progress achieved up to CM1 consisted of experimental investigations of RF-based drone detection and classification systems. Both the FFT and STFT representations of drone signals were explored and their suitability for learning based approaches was evaluated. Based on the STFT data obtained from the RFUAV dataset, a YOLOv5 model was trained to use as a baseline for subsequent experimentation with alternative CNN architectures. In parallel, FFT generation from raw data provided by the RFUAV dataset was also carried out to assess the trade-offs between FFT and STFT-based pipelines.

5.2.3.2 Progress Between CM1 and CM2

Following CM1, the progress until CM2 focused on the development of the FFT and STFT pipelines within WP1 and the construction of GNU Radio flowcharts in the context of WP2.

Initially, a numerical algorithm for bandwidth and center frequency estimation was developed. An FFT-based pipeline was established with the primary aim of detection of drones while also differentiating them from co-band signals, such as Wi-Fi or Bluetooth. This pipeline included data preprocessing, labeling and train-validation splitting. A baseline CNN model achieved stable performance. With the objective of generating consistent and visually interpretable spectrogram representations from raw IQ data, the STFT pipeline was refined. Several STFT parameter configurations were investigated. SNR estimation was also performed using power spectral density analysis derived from STFT outputs.

Progress in WP2 up to CM2 focused on preliminary setup and familiarization with GNU Radio. First, simulated signal sources were used to construct the initial flowchart. After the SDR hardware was obtained, flowchart design for the detection of real-time signals was initiated. The spectrum and spectrogram of the raw data from the RFUAV dataset and real-time captures were successfully plotted. The visualization of the signals also included the results of SNR estimation and the detection algorithm based on numerical methods.

5.2.3.3 Progress between CM2 and CM3

After CM2, the progress between CM2 and CM3 focused on the finalization of the FFT and STFT pipelines within WP1, collection of drone signals within WP2, and beginning the development of the GUI within WP3.

Following the advancements in the FFT pipeline in CM2, the FFT pipeline was finalized CM3 phase. The pipeline automatically processes the selected drone and Wi-Fi signals, divides it into frames of 4096, takes the log-magnitude spectrum, and normalizes the data. The data is then automatically labeled. The labels are then checked using the previously trained models. If the labels do not match with the results, the data is removed from the dataset.

This way contamination in the dataset is minimized. The data is then formed into matrices along with the label matrices to be used for training. The false drone positives were collected during field testing to enhance the dataset in critical areas. The numeric signal parameter extraction algorithm was also improved to detect the center frequency, bandwidth, and SNR of all signals in the current spectrum. The complete pipeline is shown in Fig. 9

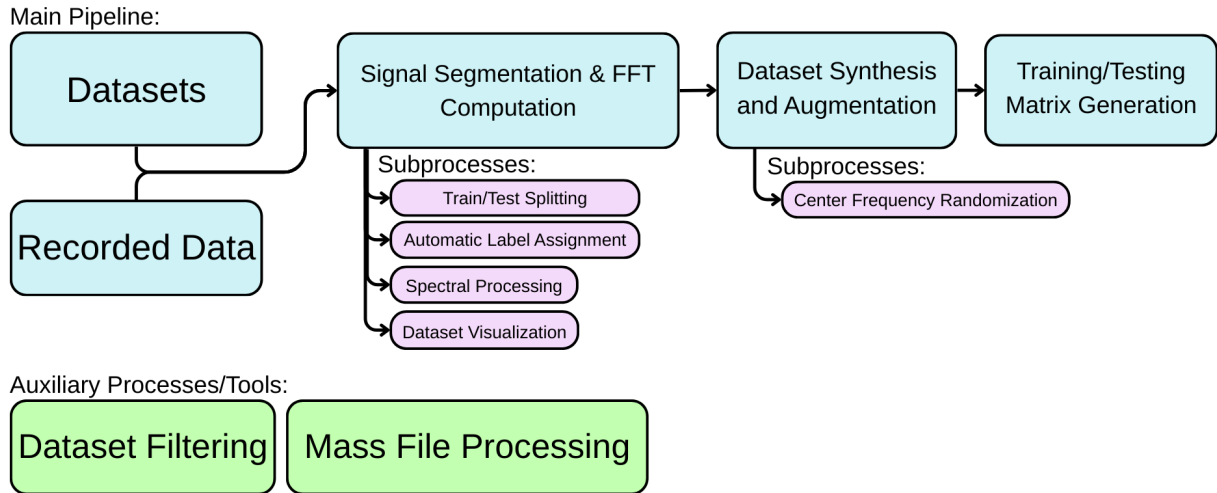


Figure 9: FFT-Pipeline Process Flowchart

The STFT pipeline was also finalized between CM2 and CM3. The spectrogram representation enables the use of deep convolutional neural networks designed for image classification and allows the model to capture both temporal and spectral characteristics of RF signals simultaneously. This pipeline converts the IQ data into segments of 30 ms. Each segment is processed using a 2048 point STFT with 75% overlap. These images are then further processed through normalization and color mapping steps. Additionally, raw data from RFUAV was downsampled from 100 MHz to 40 MHz to match with self-collected data. The STFT pipeline is shown in Fig. 10

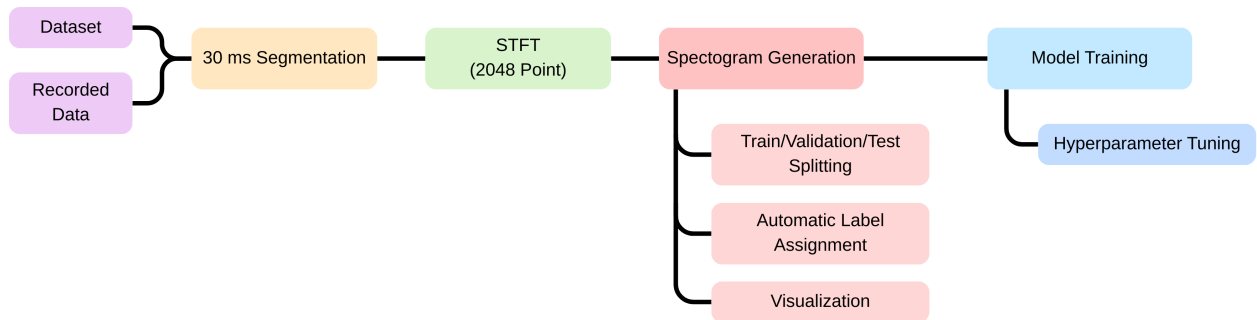


Figure 10: STFT-Pipeline Process Flowchart

Progress in WP2 between CM2 and CM3 focused on data acquisition. The B200Mini SDR was obtained from Meteksan which allowed for operation in the 2.4 GHz and 5.8 GHz bands which were previously inaccessible. Data was collected for different drones in different environments to create a rich dataset along with Wi-Fi and Bluetooth recordings.

Progress in WP3 began shortly after CM2 and saw significant improvements up to CM3. The development of the GUI was conducted using the PyQt5 library in Python along with the PyQtGraph library for real-time visualization. The application is divided into two main components: frontend and backend. Frontend runs on the main thread and is responsible for the visualization of the data and displaying the parameters. Backend is responsible for data acquisition, signal processing, deep-learning inference, and thread management. The GUI was capable of displaying the FFT and STFT of the obtained IQ data in real-time along with the detection and classification results. The application successfully implemented the signal parameter extraction algorithm to obtain and display the center frequency, bandwidth and SNR of each detected emitter which were also conveyed to the user. It featured a third plot to show the detection results along with its respective FFT. The application offered the option to switch the STFT plot to the last STFT which was inputted into the classification model, to be able to follow the decisions made by the model. The GUI of CM3 is shown in Fig. 46.

5.2.3.4 Progress between CM3 and CM4

5.2.3.4.1 Progress in WP1

FFT-Based Pipeline

Due to the automatic labeling of spectrum data according to the type of recordings, significant dataset contamination occurred. The drone signal recordings with interference also had frames where only Wi-Fi was present, the automatic labeling system also labeled some pure Wi-Fi signals as Drone. A dataset augmentation step was introduced to the FFT pre-processing pipeline. The step checks each spectrum labeled as Drone, then feeds it to the most accurate model at the time of processing. If the model labels it as Wi-Fi, the sample is deleted in order to eliminate contamination. This step in preprocessing, without any changes to the model, increased accuracy by 1.3% getting the accuracy up to 99.8%. The final confusion matrix, along with the accuracy and latency comparison for different sized networks is shown in Appendix 6.

Using the improved signal parameter extraction algorithm, the multiple detection algorithm was developed. This algorithm isolated signals using numeric methods and classified the isolated signals using the FFT-Based CNN. Similar progress was made for the FFT-based custom segmentation network SegNet which is discussed in detail in Appendix 5.

GPU-Accelerated Training Pipeline

The FFT-based CNN training workflow was migrated from a local CPU-bound Python script to a GPU-accelerated Colab notebook, with several improvements introduced to enhance generalization, training stability, and deployability. A Gaussian noise augmentation layer and label smoothing were added to improve robustness in crowded ISM bands where Wi-Fi and Bluetooth interference are prevalent. The reactive `ReduceLROnPlateau` learning rate

scheduler was replaced with a warmup followed by cosine annealing schedule, which stabilizes early training and helps the optimizer escape local minima through periodic restarts. Mixed-precision training (float16 compute with float32 master weights) and `tf.data` prefetching were enabled, significantly reducing the training time on Colab T4 GPUs. Finally, a TFLite export pipeline was added for potential edge deployment; the quantized model occupies only 116 KB compared to approximately 700 KB for the full Keras model. A comparative study of the model with and without the Gaussian noise layer was also conducted, confirming a marginal ($\sim 0.2\%$) accuracy improvement with no measurable impact on inference latency.

STFT-Based Classification Pipeline

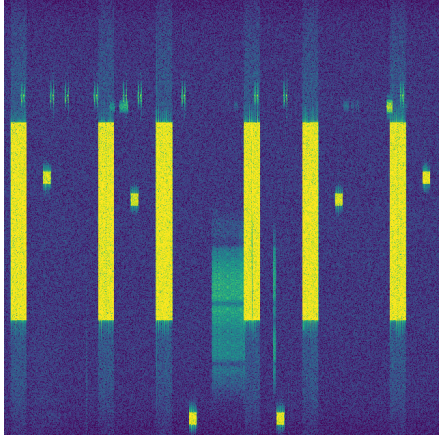
1. STFT Optimization

Building on the previously finalized STFT-based classification pipeline, the final phase of WP1 focused on improving the computational efficiency of the system, updating the dataset, training multiple deep learning models, and preparing these models for GUI-based field testing. Since the main FFT and STFT pipelines had already been developed and validated in the previous phases, this stage did not aim to redesign the complete preprocessing chain from the beginning. Instead, the main objective was to bring the STFT-based classification pipeline closer to a deployable real-time system by reducing its processing time, increasing the diversity of the training data, and comparing different model architectures under the same experimental conditions.

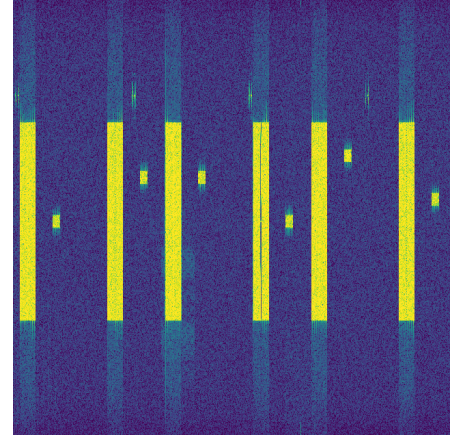
A major improvement in this phase was the reduction of the STFT size from 2048 points to 1024 points. In the previous version of the pipeline, 2048-point STFTs were used to obtain detailed time-frequency representations of the RF signals. Although this representation was useful for visualizing signal characteristics, STFT generation is computationally expensive, especially when it is repeated continuously inside a real-time GUI pipeline. During GUI integration, the total processing time of the complete pipeline could reach approximately 4 seconds in some cases. This delay was considered too high for the real-time operation requirement of the system, since the user interface should update the signal representation and classification result within a shorter time interval.

For this reason, the complete STFT image dataset was regenerated using 1024-point STFTs instead of 2048-point STFTs. The general structure of the preprocessing pipeline was preserved, including IQ segmentation, STFT generation, spectrogram image formation, normalization, color mapping, and dataset organization. However, by reducing the STFT size, the computational load of the preprocessing stage was significantly decreased. After this modification, the total runtime of the complete GUI pipeline was reduced to approximately 1.8–2.1 seconds. Therefore, this update directly contributed to satisfying the real-time timing requirement of the project. In addition, regenerating the dataset with the same 1024-point STFT configuration ensured that the training data and the GUI input representation re-

mained consistent with each other.



((a)) 2048-point STFT



((b)) 1024-point STFT

Figure 11: Comparison of spectrograms generated from the same RF signal sample using 2048-point and 1024-point STFTs. The main time-frequency characteristics are preserved in the 1024-point representation, while the computational cost is reduced.

To visually verify the effect of this change, example spectrograms generated from the same RF signal sample using 2048-point and 1024-point STFTs were compared. As shown in Fig. 11, the main time-frequency structures of the signal were preserved after reducing the STFT size. The dice score and the SSIM was also measured for these images in order to compare the loss of information. Their values were obtained as 0.95 and 0.8617 respectively. Therefore, the 1024-point STFT representation was considered sufficient for classification while providing a significant improvement in processing time.

2. Dataset Expansion with Additional Recordings

After the STFT optimization, the existing 6-class DJI dataset was updated. The six classes used in the main classification task were DJI Avata 2, DJI FPV Combo, DJI Mavic 3 Pro, DJI Mavic Air 2, DJI Mini 3, and DJI Mini 4 Pro. In addition to the previously prepared recordings, several newly collected samples were added to the corresponding class folders. These additional recordings were included to improve the diversity of the dataset and to make the training set more representative of the signals observed during practical measurements. Then, the complete 6-class dataset was regenerated using the updated 1024-point STFT configuration. This updated dataset was used as the common dataset for the final 6-class model training and comparison experiments.

3. Exploration and Assessment of Alternative CNNs and Vision Transformers

The main model of the system was selected as ResNet18. ResNet18 was preferred because it provides a strong balance between classification performance and computational complexity. In this project, the final model should not only achieve high offline accuracy, but should also be suitable for real-time GUI integration and field testing. Very large models may provide high accuracy in offline test conditions, but they can increase model size, memory usage, and inference time. On the other hand, very small models may be computationally efficient, but

they may not always provide sufficient robustness under practical field conditions. ResNet18 was therefore selected as the main model because it offers a reasonable compromise between model capacity, inference cost, and expected real-time performance. This choice was also supported by the RFUAV study, where ResNet18 and ResNet34 were reported as strong candidates for RF spectrogram-based classification [29].

However, the final phase of the project did not rely only on the previously reported results in the literature. In order to verify the suitability of the selected main model using our own updated dataset, several alternative model architectures were also trained and compared. All 6-class models were trained using the same updated 1024-point STFT training dataset and evaluated using the same test dataset. This was important for a fair comparison, because differences in model performance should mainly come from the model architecture rather than differences in the dataset or preprocessing procedure.

In addition to the main ResNet18 model, ResNet34, ResNet50, and ResNet101 were trained as deeper residual CNN alternatives. These models were included to investigate whether increasing the depth of the residual network could improve classification performance on RF spectrogram images. ResNet models are suitable for this task because their residual connections help the network learn deeper feature representations while reducing the risk of vanishing gradients. In the context of RF spectrogram classification, these models can learn local time-frequency patterns, spectral shapes, and signal structures that are useful for distinguishing different drone models. By training ResNet34, ResNet50, and ResNet101, the project evaluated whether deeper CNN architectures provide a meaningful advantage over ResNet18, or whether the additional computational cost is unnecessary for the current 6-class problem.

Transformer-based architectures were also investigated in this phase. For this purpose, SwinV2-Tiny Transformer, ViT-B/16, and ViT-L/16 models were trained. Here, ViT-B/16 refers to the Vision Transformer Base model with 16×16 image patches, while ViT-L/16 refers to the Vision Transformer Large model with 16×16 image patches. Vision Transformer models process images differently from classical CNNs. Instead of relying mainly on local convolutional filters, they divide the image into patches and use attention mechanisms to learn relationships between different regions of the image. This can be useful for spectrogram classification because some discriminative features may not be limited to a small local region; they may appear as wider time-frequency structures across the spectrogram. SwinV2-Tiny was included as a more efficient transformer-based model, while ViT-B/16 and ViT-L/16 were included to test larger attention-based models. These models allowed us to observe whether transformer-based image classification methods can provide an advantage over CNN-based models for RF spectrogram inputs.

A lightweight model, MobileNetV3-Small, was also trained. MobileNetV3-Small is designed for efficient inference and low computational cost. It is especially useful in real-time or resource-limited applications where fast response is important. Since the final drone detection system is intended to operate inside a GUI with real-time signal processing and visualization, lightweight models are important candidates. The purpose of training MobileNetV3-Small was to examine whether a smaller model could provide acceptable classification performance while reducing the computational burden of the system. However, the field-test observations showed that being lightweight alone is not sufficient for this project. Although MobileNetV3-Small had the smallest model size among the trained models, its GUI-integrated field-test accuracy was significantly lower than expected. Therefore, this result showed that the final model must be both lightweight enough for real-time use and robust enough for practical RF signal conditions.

Another important motivation for training different model families was the large difference in their model sizes and deployment requirements. The trained models covered a wide range of sizes, from very small mobile-oriented architectures to large transformer-based models. This diversity was intentional because the project requires a model that can provide high accuracy, stable field-test behavior, and reasonable computational cost at the same time. The model sizes of the trained 6-class architectures are summarized in Table 5. In addition, the confusion matrices of all trained classification models are provided in Appendix 7 for a more detailed class-based performance analysis.

Table 5: Model size comparison of the trained 6-class architectures.

Model	Model Size
MobileNetV3-Small	4.8 MB
ResNet18	42.7 MB
ResNet34	81.3 MB
ResNet50	90 MB
SwinV2-Tiny Transformer	108 MB
ResNet101	162 MB
ViT-B/16	327 MB
ViT-L/16	1.12 GB

As seen from Table 5, the trained models have very different storage and deployment costs. Larger models such as ViT-L/16 and ViT-B/16 require significantly more memory, while MobileNetV3-Small has a very small model size. However, the experiments showed that model size alone cannot determine the best model for the system. A larger model does not always provide a meaningful advantage for the current task, especially when its accuracy is close to smaller models but its memory and computational requirements are much higher. Similarly, a very lightweight model is not automatically the best deployment choice if it cannot maintain reliable classification performance under field conditions. Therefore, the

comparison was not limited to accuracy values alone. Model size, memory requirement, real-time suitability, GUI integration behavior, and field-test stability were also considered as important factors for selecting a practical model for the final system.

Therefore, the training of multiple model architectures served two purposes. First, it provided a systematic comparison between different model families, including residual CNNs, transformer-based models, and lightweight mobile models. Second, it allowed the robustness of the main ResNet18-based choice to be tested against alternative architectures. Since the current 6-class classification problem contains a limited number of drone classes, most models were expected to achieve high offline accuracy. For this reason, offline accuracy alone was not considered sufficient for final model selection. Instead, computational complexity, model size, GUI integration behavior, and field-test performance were also considered important evaluation criteria. This approach is more consistent with the final goal of the project, which is not only to train an accurate model, but also to develop a reliable and deployable RF-based drone detection and classification system.

4. Ablation Study of ResNet18

Additionally, an ablation study was conducted to evaluate the contribution of different architectural components to the overall classification performance of ResNet18. The four models used for this study include the original baseline ResNet18 architecture, a version containing only the Squeeze-and-Excitation (SE) attention block (SE Only), a version containing only the redesigned fully connected classifier head (Custom FC Only), and the final proposed model combining both modifications (Full Proposed Model) [30, 31]. The ablation process required training and evaluating four separate deep learning configurations independently while preserving identical preprocessing, dataset composition, optimization parameters to ensure a fair comparison between the architectural variants. Each model underwent a complete training for 10 epochs, validation, and testing cycle, including parameter analysis, confidence evaluation, forward-pass latency measurement, and model size estimation. A detailed summary of the performance of these four models comparing accuracy, F1 score, forward pass inference latency, and model size can be found in Appendix 9.

While the other three models achieved a perfect F1 score of 1.000, only Custom FC Only achieved a slightly lower F1 score of 0.9946. Even though the performance degradation is small, this result suggests that the redesigned classifier head alone is insufficient to fully preserve the feature discrimination capability of the original architecture of ResNet18. We can conclude from the performance metrics that the SE attention mechanism contributes positively to feature refinement and model stability, whereas the combination of both architectural modifications provides the most balanced overall configuration in terms of classification accuracy, confidence consistency, and computational performance.

5. Multi-Class Expansion of ResNet18

In addition to the 6-class model comparison, an extended 9-class dataset was also prepared to evaluate the scalability of the proposed pipeline. For this purpose, three additional UAV-related classes were selected from the RFUAV dataset and added to the existing 6-class dataset. These additional classes were Autel EVO Nano, Herelink HX4, and Yunzhuo H16. The previously generated 6-class 1024-point STFT dataset was copied and extended with these three new classes. Since the additional samples were obtained from the RFUAV dataset, the corresponding IQ recordings were first downsampled and then processed using the same 1024-point STFT-based preprocessing structure. After generating the spectrogram images, the extended dataset was organized into training, validation, and test sets. A separate 9-class model was then trained to observe whether the developed pipeline could be extended to additional UAV-related classes without requiring a fundamental redesign.

Representative STFT examples of the three additional UAV-related classes are shown in Fig. 12. These examples illustrate the visual time-frequency characteristics of the newly added classes after the same preprocessing and spectrogram generation steps were applied with the same STFT parameters (1024 STFT points, 30 ms segmentation, 75% overlap).

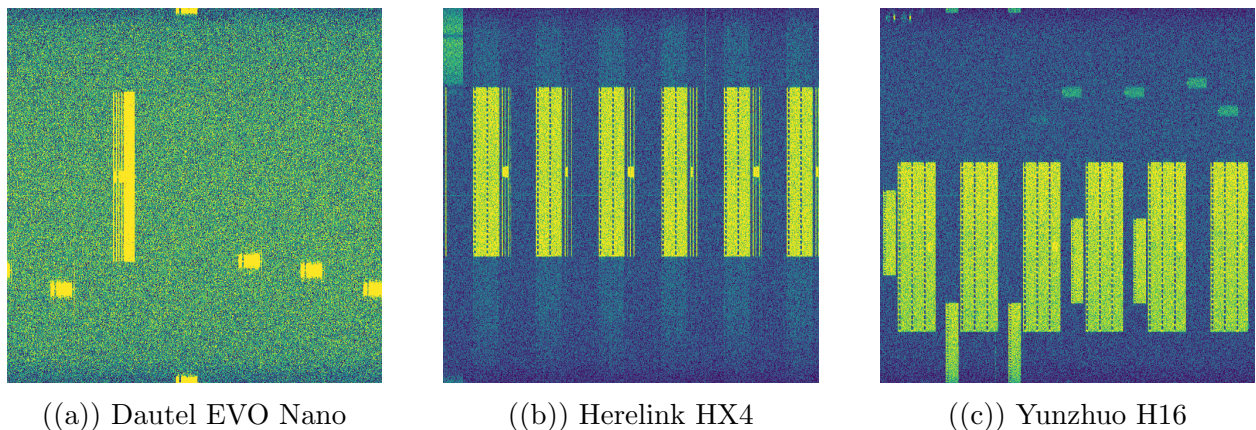


Figure 12: Representative STFT spectrogram examples of the three additional UAV-related classes included in the extended 9-class dataset.

This 9-class experiment was important because the project is designed with a modular structure. In a practical system, new drone models may need to be added to the dataset over time. Therefore, the preprocessing, dataset generation, training, and testing stages should be reusable when new classes are introduced. The extended dataset experiment showed that the STFT-based pipeline can be adapted to additional UAV-related classes by following the same processing and training procedure. Although the detailed numerical results of the 9-class model are presented in Chapter 6, this experiment provided an important progress step toward a more scalable drone detection and classification framework.

During this extension stage, additional UAV-related controller signal data from the RFUAV

dataset was also examined. The initial motivation was to make the system more general by considering not only drone model signals, but also different UAV-related RF communication signals. However, these controller signals were not directly included in the final downsampled 40 MHz training dataset. When the 100 MHz sampled RFUAV controller signals were downsampled, aliasing effects were observed in the generated STFT representations. In addition, the duty cycles of these controller signals were relatively low, which means that the signal activity was not clearly visible in every STFT segment. As a result, directly mixing these downsampled controller signals with the drone model spectrograms was not considered reliable for the final training stage. This observation is discussed in more detail in Chapter 6 as one of the limitations and future work directions of the project.

Finally, all trained models were prepared for GUI integration and field testing. This step was essential because high performance on an offline test set does not automatically guarantee stable performance in a real-time system. In offline evaluation, the model receives pre-generated and well-organized spectrogram images. In the GUI, however, the model receives spectrograms generated from live or field-recorded IQ data through the real-time processing pipeline. Therefore, the model must be robust to practical effects such as preprocessing variations, signal quality changes, environmental interference, and timing constraints. By integrating the trained models into the GUI, the project evaluated not only their offline classification capability, but also their practical usability in a real-time RF-based drone detection and classification system. The detailed numerical comparison of the models, including accuracy, precision, recall, F1-score, model size, and GUI-integrated field-test observations, is presented in Chapter 6.

CRNN Pipeline Preprocessing (FFT and STFT)

In parallel with the finalized CNN and ResNet18 pipelines, preprocessing workflows were developed for an alternative CRNN-based (Convolutional Recurrent Neural Network) architecture targeting both FFT and STFT modalities. The motivation for this direction is that, unlike a frame-wise CNN or ResNet18 which treats the spectrogram as a static 2D image, a CRNN processes the input along its time axis as a sequence, making temporal patterns such as drone control bursts and packet cadence directly learnable through stacked Bidirectional LSTM layers.

5.2.3.4.2 Progress in WP2

Data Acquisition

The dataset obtained in the CM3 phase has been enriched with indoors recordings of the models DJI Mini 3 and DJI Mini 4 Pro. Additional recordings of DJI Mavic Air 2 were taken near Odeon for more data of that model without interference. Indoors recordings feature Wi-Fi interference while outside recordings were free of such signals. Consequently, capturing of Wi-Fi and Bluetooth signals outside were infeasible and not necessary. The Wi-Fi and Bluetooth signals are emitted from multiple emitters in the environment which

provides a wide range of signals with different SNRs in a single recording with high SNR.

Table 6: Overview of All Obtained Recordings

Recording Type	Bandwidth		Environment		Center Frequency	SNR Conditions		
	10 MHz	20 MHz	Indoors	Outdoors		low	medium	high
DJI Mini 3	✓	✓	✓	✓	2.4505 GHz	✓	✓	✓
DJI Mini 4 Pro	✓	✓	✓	✓	2.4505 GHz	✓	✓	✓
DJI Mavic Air 2	✓	✓	✓	✓	2.4505 GHz	✓	✓	✓
Wi-Fi/Bluetooth	N/A	N/A	✓	-	2.4125 GHz	-	-	✓

Test Data Generation

To ensure a healthier and more reliable evaluation procedure, a completely separate test dataset was collected independently from the training and validation data. The test recordings were preserved exclusively for final performance assessment and were never used during model training, or architectural optimization stages. This separation helped prevent data leakage and ensured that the performance metrics reflected the model’s true generalization capability on previously unseen RF environments and signal conditions. The raw IQ recordings belonging to the reserved test dataset were processed independently to generate STFT-based spectrogram representations using the same preprocessing pipeline and STFT parameters applied during training. This methodology improved the capability across different drone signal instances.

Drone ID Processing

The constructed GNU Radio flowgraph is designed to intercept and demodulate DJI DroneID signals, which utilize a modified Orthogonal Frequency Division Multiplexing (OFDM) demodulation scheme. The system takes high-bandwidth I/Q data as inputs, performing demodulation, and extracting telemetry data via an external Python block. The input and pre-processing stage consists of four main parts. Firstly, the IQ data is throttled accordingly. Secondly, synchronization and correction is completed. ”Schmidl & Cox OFDM Synch” block utilizes a known preamble structure to estimate timing and frequency offsets which are then used to compensate the signal. Thirdly, demultiplexing and transformation part is completed. Lastly, extraction and decoding is completed by using Keep M in N, Constellation Decoder, and ZMQ PUSH Sink.

The integration of the Schmidl & Cox algorithm with a Python-based post-processor allows the system to overcome significant phase noise and frequency offsets inherent in high-frequency drone communications. While the Constellation Sink may display ”phase rotation” due to residual offsets, the Python parser implements Blind Phase Recovery to extract the final Serial Number and GPS Coordinates of the target UAV.

```

=====
[DJI DRONEID PARAMETRE RAPORU]
** Seri Numarası : Aranıyor...
** Enlem (Lat) : 36.776185
** Boylam (Lon) : 28.024633
** İrtifa (AGL) : 88.5 metre
=====

```

Figure 13: Drone ID Telemetry Results

The output of the system is shown in Fig.13. When the system identifies a drone, the latitude, longitude, altitude information are extracted from the signal. Drone's serial number cannot be determined as of CM4, and the accuracy of the telemetry values are to be improved. These will be pursued as part of future plans. due to signal quality. The GNU Radio design consists of the detailed and complex block design shown in Appendix 11.

5.2.3.4.3 Progress in WP3

The development of the GUI has been finalized as of CM4. This report describes the progress since CM3 and the finalized state of the application. The system is designed to acquire real-time IQ data via SDRs or read pre-recorded IQ data and process the information in real-time using numeric signal processing algorithms and the developed deep-learning models while visualizing the data. The application uses multiple threads to ensure the processes run in real-time without adding latency or disrupting the responsiveness of the algorithm. The application can be controlled by the user by modifying a configuration file or via the selections in the GUI which allow for real-time control over the application.

Frontend of the Application

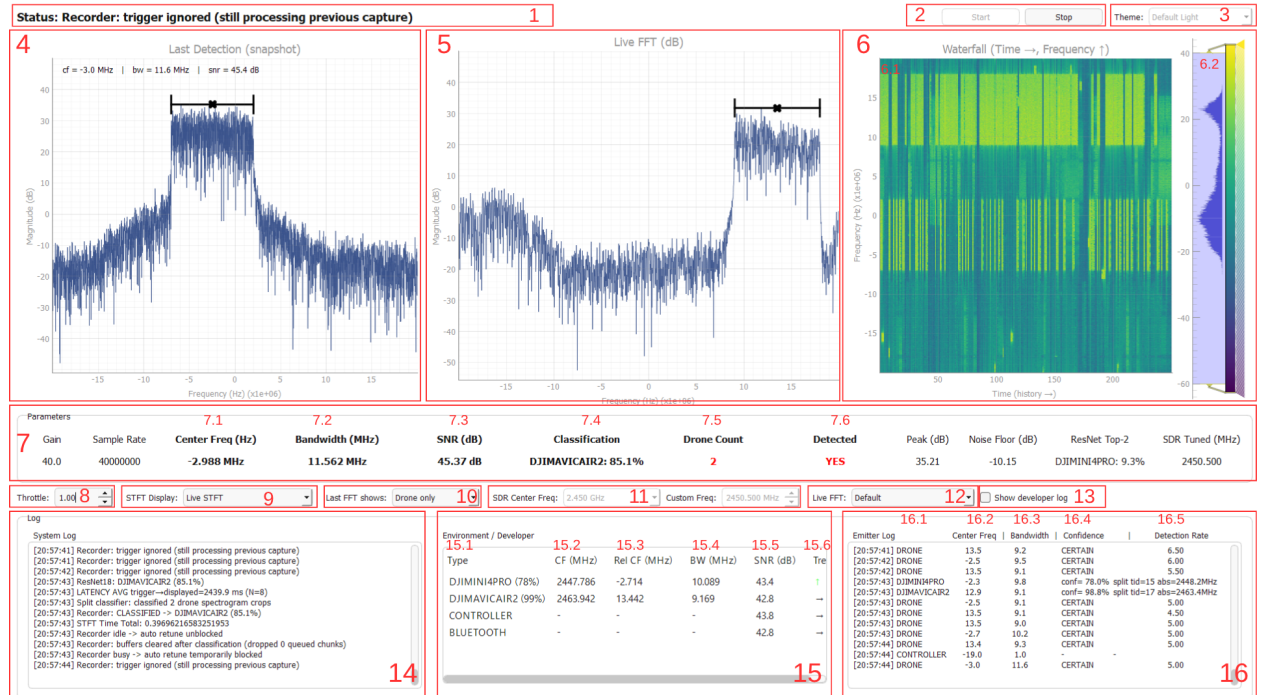


Figure 14: Annotated GUI of the Application

Building upon the preliminary GUI, the application was improved to support the newly developed multiple detection and classification algorithms, more settings and configurations. A larger version of the GUI is available in Fig. 45

(1) Status Indicator:

The first and leftmost panel seen in Fig. 14 is the status indicator. This indicator allows the user to see the status of the application and the ongoing processes. Provides information regarding IQ data source, data recording state for STFT, STFT taking and ResNet model output times, and other possible statuses moving forward.

(2) Start/Stop Controls:

The second panel in Fig. 14 is the start/stop buttons. These buttons allow the user to start and stop the signal acquisition and processing pipeline.

(3) Theme Selector:

The third panel seen in Fig. 14 is the theme selector. It allows the user to choose between different visual themes of the GUI.

(4) Last Detection Panel:

The fourth panel seen in Fig. 14, the last detection panel, displays the most recent detected signal segment. It represents a snapshot of the signal corresponding to the last confirmed detection event. The purpose of this panel is to allow the user to clearly observe the spectral characteristics of the detected emitter without being affected by continuous updates. This is particularly useful for analyzing the structure of detected drone or communication signals.

(5) Live FFT Panel:

The fifth panel in Fig. 14 is the live FFT panel. This panel shows the real-time frequency-domain representation of the incoming signal. Unlike the last detection panel, this plot is continuously updated and reflects the current RF environment. It allows the user to observe instantaneous spectral activity and detect sudden changes in the signal.

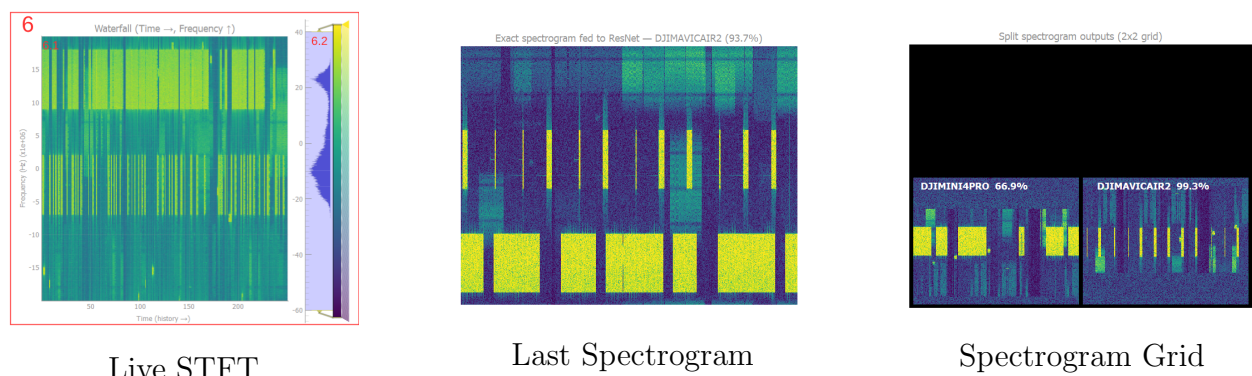


Figure 15: STFT and Spectrogram Display Panels

(6) Waterfall-STFT and ResNet Spectrograms (Time–Frequency Representation):

This part of the GUI displays the time-frequency representations of the received RF signal. On the left side, the live waterfall spectrogram is shown, and its color limits can be adjusted automatically to improve the visibility of the signal. The middle panel shows the last STFT image that is directly sent to the ResNet classification model. In multi-drone scenarios, the detected signal regions are separated and each drone signal is shifted to baseband. Then, the individual STFT images of these separated drone signals are displayed in a grid on the right side. This allows the user to observe both the live RF environment and the exact spectrogram inputs used by the classification model.

(7) Signal Parameter Panel:

The seventh panel seen in Fig. 14, the parameter tile, provides numerical information about the detected signal. It is divided into multiple sub-components:

- **(7.1) Frequency:** Displays the estimated center frequency of the detected signal.
- **(7.2) Bandwidth:** Indicates the bandwidth occupied by the signal.
- **(7.3) SNR:** Provides the estimated signal-to-noise ratio.
- **(7.4) Drone Type & Confidence:** Displays the classified drone type together with the confidence score produced by the ResNet18 model. This indicates both the predicted class of the detected drone and the reliability of the classification result in percentage.
- **(7.5) Drone Count:** Shows the number of drone signals detected in the current RF spectrum. This is especially useful in multi-drone scenarios, where more than one drone signal may be present at the same time.
- **(7.6) Drone Detection:** Indicates whether a drone signal is detected or not. This output gives a direct YES/NO result to help the user quickly understand the presence of drone activity.

These values are computed in the backend and updated in real time, providing quantitative insight into the detected signal.

(8) Throttle Control:

The eighth panel seen in Fig. 14 is the throttle selector. This control allows the user to adjust the update rate of the visualization components, particularly the STFT-based time–frequency representation. By modifying the throttle setting, the user can control how frequently new data is processed and displayed. A higher update rate results in smoother and more responsive visualizations, while a lower rate reduces computational load and allows clearer observation of slower signal variations. This provides flexibility for the user to select

an appropriate visualization speed depending on the signal characteristics and system performance.

(9) STFT Display Mode Selection:

The ninth panel seen in Fig. 14 is responsible for the STFT mode selection. This control allows the user to switch between two different STFT visualization modes. In the first mode, the GUI displays the live STFT stream in the form of a continuously updating waterfall plot, enabling real-time observation of the signal’s time–frequency evolution.

In the second mode, the GUI displays the specific STFT segment that is forwarded to the ResNet18 classification model. This provides a direct view of the exact input used by the model during inference.

In addition to these two modes, a third mode was added for multi-drone cases. In this mode, the GUI shows the separate STFT images of each detected drone signal in a grid. These images are shifted to baseband before being displayed.

This functionality improves interpretability by allowing the user to understand how classification decisions are made, as the model input can be directly inspected and compared with the live signal behavior.

(10) FFT Display Selection:

Similarly, the tenth panel seen in Fig. 14 determines the content shown in the “Last Detection” panel (1). The user can choose between displaying only the FFT of the last detected drone signal or displaying the FFTs of all detected emitters, including Wi-Fi, Bluetooth, and drone signals.

When the single-emitter mode is selected, the panel focuses on the most recent drone detection, providing a clean and focused view of the relevant signal. In contrast, the multi-emitter mode allows the user to observe and compare different detected signals within the interface.

This flexibility enables better analysis of the RF environment and helps the user understand how different emitters are represented in the frequency domain.

(11) SDR Center Frequency Selector:

The eleventh panel seen in Fig. 14 allows the user to choose the SDR center frequency. The user can select Auto, where the SDR cycles through predetermined center frequencies automatically, 2.4505 GHz, 5.1 GHz, 5.8 GHz, or enter a custom frequency manually.

(12) Live FFT Selector:

The twelfth panel seen in Fig. 14 is the live FFT selector. This selector allows the user to choose how the live FFT is displayed. The available modes are normal display, hold maximum for 10, 25, 50, or indefinite frames, and average over 10, 25, 50, or indefinite frames. This helps the user observe either instant spectrum changes or more stable FFT behavior.

(13) Developer Log Selector:

The Twelfth panel seen in Fig. 14 is the developer log selector. This checkbox allows the user to show or hide the developer log. When it is enabled, detailed backend messages and debugging information can be displayed on developerlog panel. This is useful for development and testing purposes.

(14) System Log Panel:

The fourteenth panel seen in Fig. 14 is the system log. This panel records system events, detections, and processing steps in chronological order. It includes information such as detected emitters, processing stages, and system messages. This is particularly useful for debugging and tracking system behavior over time.

(15) Developer Log and Environment Information Panels:

The fifteenth panel seen in Fig. 14 is the environment information panel. This panel gives a summary of the detected RF environment. It is shown on the lower-right side of the GUI when the application is first opened.

- **(15.1) Type:** Shows the detected signal or emitter type, such as a drone, controller, or Bluetooth signal.
- **(15.2) CF:** Displays the estimated center frequency of the detected signal in MHz.
- **(15.3) Rel CF:** Shows the relative center frequency of the signal with respect to the SDR center frequency.
- **(15.4) BW:** Displays the estimated bandwidth of the detected signal in MHz.
- **(15.5) SNR:** Shows the estimated signal-to-noise ratio of the detected signal.
- **(15.6) Interest:** Shows the trend of the SNR value. It indicates whether the SNR is increasing, decreasing, or remaining stable.

When the *Show developer log* checkbox is selected, the environment information panel is replaced by the developer log panel. In this mode, backend messages and debugging information are shown instead of the environment table.

(16) Emitter Log Panel:

The sixteenth and the last panel seen in Fig. 14 is the emitter log. This panel presents the

outputs of the detection and classification stages together with key signal parameters. It is organized into multiple sub-components:

- **(16.1) Emitter Type:** Displays the detected emitter category, such as Drone, Controller, Wi-Fi, or Bluetooth.
- **(16.2) Center Frequency (f_c):** Indicates the estimated center frequency of the detected signal.
- **(16.3) Bandwidth (BW):** Shows the estimated bandwidth of the signal.
- **(16.4) Confidence and Certainty:** Displays the confidence score of the classification model together with the detection certainty. The confidence value corresponds to the output of the ResNet18 model, while the certainty value reflects the reliability of the detection stage.
- **(16.5) Drone Detection Rate:** Represents the proportion of time in the last two seconds during which a drone signal has been detected. This provides a short-term temporal indication of drone presence.

Backend of the Application

The frontend is designed to operate asynchronously with the backend processes, ensuring that visualization updates do not interfere with real-time signal processing. This separation allows the system to maintain responsiveness while continuously processing incoming RF data. In the background of the application, several other processes run in different threads:

UnifiedIQWorker: This worker, reads IQ data from a given file or acquires the data from a real-time SDR input. These samples are then transferred to the separate processes that process the data. This process is also responsible for the timing of the samples and throttling. Also sets the parameters of the SDR like center frequency, bandwidth or the gain.

SpectralWorker: This is an intermediate process that processes the IQ data by applying different windows, taking the FFT, and forwards it to be visualised or fed into the detection algorithm.

DetectorWorker: This process is responsible for swift information extraction from the spectra. This process makes the necessary preprocessing steps to prepare the data to be input into the FFT-Based CNN. The signal is also processed by the numeric signal parameter extraction algorithm to make a decision based on the outputs of both systems. The signals are then identified as one of 5 classes: "Bluetooth", "Controller" (which refers to the remote that is controlling the drone), "Drone", "Wi-Fi", and "Atmospheric Noise". The signal parameters such as SNR, bandwidth, and center frequency are also extracted in the processes and all are sent into the main thread to be displayed to the input.

MultipleDetectorWorker: The MultipleDetectorWorker records 8 ms of IQ data at once and the data is then fed to the multiple detection algorithm. This ensures all the emitters in the environment are detected without missing any. However, due to this precision, it falls behind the main DetectorWorker. If multiple drones' signals are not detected at the same time frame, consistent detections at different center frequencies are also flagged as multiple drones.

RecorderWorker: This process records the data for a fixed amount of time as. The data is accumulated in a buffer to ensure no consecutive IQ data frames are dropped. This enables the collection of a continuous IQ data stream which then allows us to take the Short-Time Fourier Transform to observe the time-frequency distribution of the signal. The spectrogram is then processed by the multiple classification algorithm to prepare each signal individually for classification. These images are fed to the STFT-Based ResNet Model to obtain a classification result.

MainWindow: This thread is responsible for processing inter-thread communications and signals, starting and stopping other threads, sending the user inputs to their respective workers, and displaying the data to the user. The data flow in the process is as follows:

- Firstly, the IQ data is obtained from the UnifiedIQWorker is forwarded to SpectralWorker. SpectralWorker's output spectra are sent to the other workers that process IQ data. The spectra are also plotted as "Live FFT"s and concatenated in time to obtain a direct time-frequency distribution which is equivalent to a 0% overlap STFT that is displayed to the user in real time.
- If the DetectorWorker detects an emitter, the last detected plot is updated accordingly with the spectrum that corresponds to that emitter's spectrum. The detection outputs are then filtered using a temporal consistency filter to ensure fluctuations and temporary false positives do not get conveyed to the user. After a certain detection, the RecorderWorker gets triggered which then records the data 5ms before the detection and 25ms after the detection leading to a 30ms recording in total. This recording's STFT is taken and the ResNet model provides the classification results.
- The MultipleDetectorWorker takes burst recordings of IQ data to reliably and precisely detect each emitter in the environment without missing any shortlived signals due to long processing times. The extracted center frequencies are then used by the RecorderWorker to isolate each drone signal in the spectrogram and classify each drone separately. This multiple detection and classification information are then presented to the user in the Environment Tab in the interface.
- Finally, all of these parameters and plots are displayed to the user. In order to minimize the updates, only rising edges of consecutive identical emitter detections or rising edges of certain drone detections are conveyed to the user. The information about the emitters are

then logged to a specific emitter log and the last detected emitter plots are also only updated at rising edges.

5.2.4 Encountered Risks

- GNU Radio was feared to lose real-time operation when using python blocks. Instead of implementing the algorithms and models into GNU radio, as a precaution we looked into different ways of interfacing Python scripts with the SDR. The interface between GNU Radio and the Python environment was implemented directly to the GUI itself using the Radioconda library.
- False positivies of the detection algorithm could have costly consequences. We looked into different algoithms to prevent false positives. Finally, the temporal consistency algorithm was developed which consistently prevents false positives.
- The ResNet model could have been overfitting certain patterns of the drone’s communication signals which are may be absent in other environments. We looked into dataset balancing or modifying, model tuning etc. The model successfully generalized to different signals without requiring interventions.
- The multiple detection and classification algorithms utilise artificial signal supression and noise generation. The models could have struggled with these modified images as the artificial signals were not present in the training dataset. To resolve this, the dataset was enriched with artificially modified data.
- The CRNN-based classification pipeline, for which the FFT and STFT preprocessing workflows have already been finalized, will be completed by training and evaluating CRNN architectures (TimeDistributed CNN encoder with stacked Bidirectional LSTM layers) on both modalities.

6 Results, Discussions, and Future Directions

6.1 Results

The developed system was tested in different scenarios for different metrics such as detection and classification both for offline and real-life scenarios, signal parameter accuracy, process durations and latencies. The effectiveness of the finalized system has been verified in account of the project requirements.

6.1.1 Timing Related Results

Each process’s average latencies were given in Table 7. Detailed duration and latency breakdowns for all workers are available in Appendix 13. Other detailed timing related results are also available in Appendix 13.

Table 7: End-to-End Average Latency of the Main Processing Workers

Worker	End-to-End Average Latency (ms)
Feed→Classification	1818.8
Feed→Detection	127.43
Feed→Multiple Detection	2849.00

6.1.2 Detection Accuracy Results

The false decisions made by the detection system were also measured during real-life tests. Three different statistics were measured. Internal false positives, false positive decisions that were suppressed by the Temporal Consistency Filtering (TCF) algorithm (see section 5.2.2.1.1). False positive alerts issued, false positives that could not be suppressed by the TCF algorithm. Critical False Negatives, instances where drones were present yet the system did not detect any drones. The number of these events during a 8-hour session in a crowded Wi-Fi environment were 280, 1, and 0 respectively, which are also shown in Table 8.

Table 8: Error Rate for Detection Systems

Type	Average Occurences per Hour
Internal False Positives	35
False Positive Alerts Issued	0.1
Critical False Negative	0

The recall, precision, and f1-score were also calculated for the FFT-Based model and the results are provided in Table 9 with an overall accuracy of 99.78%.

Table 9: Classification Performance Metrics Obtained for the FFT-Based Model

Class	TP	FP	FN	TN	Precision	Recall	F1-score
Noise	2400	0	0	3972	100.00%	100.00%	100.00%
Drone	2203	5	9	4155	99.77%	99.59%	99.68%
Wi-Fi	1755	9	5	4603	99.49%	99.72%	99.60%

6.1.3 Classification Related Results

The classification-related results are evaluated in this section by considering both offline test performance and deployment suitability. Since the final objective of the project is not only to train a high-accuracy model, but also to use the model inside a real-time GUI-based RF drone detection system, the evaluation was not limited to accuracy alone. Model size, computational cost, GUI integration behavior, confidence stability, and field-test performance

were also considered as important criteria. Therefore, the trained models were compared from both machine learning and real-time system perspectives.

All 6-class models were trained using the same updated 1024-point STFT dataset and evaluated on the same test dataset to provide a standardised baseline. This setup ensured that the differences mainly came from the model architectures. The six classes used in this experiment were DJI Avata 2, DJI FPV Combo, DJI Mavic 3 Pro, DJI Mavic Air 2, DJI Mini 3, and DJI Mini 4 Pro. The tested models included residual CNN-based architectures, transformer-based architectures, and a lightweight mobile-oriented architecture. This model diversity was intentional because the final system requires a model that is both sufficiently robust and sufficiently lightweight for real-time use.

The overall offline test results of the 6-class models are given in Table 10. Macro-averaged precision, recall, and F1-score represent the average class performance by treating each class equally, while the weighted F1-score also considers the number of samples in each class. Therefore, these metrics provide a more complete view than accuracy alone.

Table 10: Offline classification results of the trained 6-class models using the same 1024-point STFT test dataset.

Model	Model Size	Accuracy	Macro Precision	Macro Recall	Macro F1-score	Weighted F1-score
ResNet18	42.7 MB	1.0000	1.0000	1.0000	1.0000	1.0000
ResNet34	81.3 MB	0.9988	0.9987	0.9996	0.9992	0.9988
ResNet50	90 MB	0.9982	0.9981	0.9994	0.9988	0.9982
ResNet101	162 MB	0.9780	0.9792	0.9916	0.9846	0.9786
SwinV2-Tiny Transformer	108 MB	0.9881	0.9882	0.9953	0.9916	0.9882
ViT-B/16	327 MB	0.9988	0.9987	0.9994	0.9990	0.9988
ViT-L/16	1.12 GB	0.9715	0.9739	0.9783	0.9743	0.9717
MobileNetV3-Small	4.8 MB	0.9970	0.9906	0.9990	0.9947	0.9971

As shown in Table 10, most of the trained models achieved very high offline classification performance on the 6-class test dataset. ResNet18 achieved perfect accuracy, precision, recall, and F1-score under offline test conditions. ResNet34, ResNet50, and ViT-B/16 also achieved very high results, with accuracy values above 0.998. MobileNetV3-Small also achieved a high offline accuracy of 0.9970 despite having the smallest model size. These results show that the updated 1024-point STFT representation preserved discriminative time-frequency information for the 6-class classification problem.

However, the results also show that a larger model does not automatically provide better performance. For example, ViT-L/16 was the largest model with a size of 1.12 GB, but it achieved lower accuracy than ResNet18, ResNet34, ResNet50, ViT-B/16, and MobileNetV3-Small. Similarly, ResNet101 was larger than ResNet18, ResNet34, and ResNet50, but its offline accuracy was lower than these smaller residual models. This indicates that increasing the depth or size of the architecture does not necessarily improve the classification perfor-

mance for this dataset. In the current 6-class case, smaller and medium-sized models were able to learn the required STFT-based features effectively.

The lightweight model results should also be interpreted carefully. MobileNetV3-Small had the smallest model size, only 4.8 MB, and it achieved high offline accuracy. At first, this result may suggest that MobileNetV3-Small is a very suitable model for real-time deployment. However, GUI-integrated field-test observations showed that its real-time performance was significantly weaker than expected. Therefore, being lightweight alone is not sufficient for this project. The final model must be lightweight enough to satisfy real-time processing requirements, but it must also be robust enough to work under practical RF signal conditions. This is one of the main reasons why offline accuracy and model size should be evaluated together with field-test behavior.

The class-based results also provide useful observations. For most models, DJI Avata 2, DJI FPV Combo, DJI Mavic 3 Pro, and DJI Mini 4 Pro were classified with very high precision and recall. The main performance drops were generally observed around DJI Mavic Air 2, DJI Mini 3, and DJI Mini 4 Pro for some models. This is reasonable because these drone models may produce more similar RF time-frequency patterns compared to more distinct classes. For example, ResNet101 achieved lower performance mainly due to the reduced recall of DJI Mavic Air 2 and the reduced precision of DJI Mini 3. ViT-L/16 also showed a clear performance drop for DJI Mini 4 Pro. These results indicate that the most difficult cases are not necessarily related to all classes equally, but mostly to drone models with more similar spectrogram characteristics.

In addition to the 6-class comparison, an extended 9-class experiment was also conducted to test the scalability of the proposed STFT-based pipeline. For this experiment, three UAV-related classes from the RFUAV dataset were added to the existing 6-class dataset. These additional classes were Autel EVO Nano, Herelink HX4, and Yunzhuo H16. The purpose of this experiment was to observe whether the dataset generation, preprocessing, and training structure could be extended to additional UAV-related classes without requiring a fundamental redesign. The overall result of the 9-class ResNet18 model is given in Table 11.

Table 11: Offline classification result of the extended 9-class ResNet18 model.

Model	Accuracy	Macro Prec.	Macro Rec.	Macro F1	Weighted F1
ResNet18 9-class	0.9994	0.9996	0.9999	0.9997	0.9994

The 9-class ResNet18 model achieved an accuracy of 0.9994, with a macro F1-score of 0.9997 and a weighted F1-score of 0.9994. This result is important because it shows that the proposed STFT-based classification pipeline can be extended to additional UAV-related classes while maintaining high offline classification performance. Since the project is designed as a modular system, new drone or UAV-related classes can be added by following the same

preprocessing, STFT generation, dataset organization, and model training procedure. Therefore, the 9-class experiment supports the scalability of the developed classification framework.

Although the offline results are very strong, they should not be interpreted as the only indicator of final system performance. In offline testing, the models receive pre-generated and well-organized STFT images. In the GUI, however, the STFT images are generated from real or field-recorded IQ data through the real-time processing pipeline. Therefore, the model input can be affected by signal strength, interference, frequency offset, low-SNR regions, window selection, and preprocessing variations. Because of this difference, a model with very high offline accuracy may still show unstable behavior under real-time field conditions.

For this reason, all trained models were also integrated into the GUI for field-test evaluation. This step was necessary to understand whether the models could preserve their classification performance when used inside the real-time system. The field-test results will be evaluated using not only the correctness of the prediction, but also the average confidence, pipeline time, and prediction stability. Table 12 is prepared for the model-based field-test comparison. The numerical values will be completed after the final field-test logs are organized.

Table 12: GUI-integrated field-test performance of the trained models.

Model	Field-Test Accuracy	Average Confidence	Average Pipeline Time (s)	Observation
ResNet18	98.83%	98.7%	1.5	Main candidate model
ResNet34	98.91%	98.8%	1.6	Strong CNN alternative
ResNet50	99.01%	98.6%	2.1	Deeper CNN model
ResNet101	98.97%	98.8%	2.75	Larger CNN model
SwinV2-Tiny Transformer	98.1%	99.1%	3.6	Transformer-based model
ViT-B/16	80.1%	95.1%	1.4	Transformer base model
ViT-L/16	98.0%	99.1%	3.3	Very large transformer model
MobileNetV3-Small	10.1%	20.1%	1.5	Lightweight but weak field-test behavior

The expected difference between offline and GUI-integrated results is an important deployment issue. This difference does not necessarily mean that the offline models were trained incorrectly. Instead, it reflects the gap between controlled dataset evaluation and real-time RF operation. In offline testing, the STFT images are generated under fixed and repeatable conditions. In the GUI, each prediction depends on the selected time window, the quality of the received IQ samples, the strength of the drone signal, and the presence of other signals in the same frequency band. Therefore, real-time predictions can become less stable than offline predictions.

One possible improvement for future GUI operation is to avoid making the final decision from only a single STFT frame. Instead, several consecutive predictions can be combined using majority voting or confidence averaging. This would reduce the effect of weak or unrepresentative signal windows and improve the stability of the final decision. Such a strategy

is especially useful when the drone signal is intermittent, weak, or partially affected by interference.

The comparison also shows why the final model selection should be based on a balanced trade-off. A very large model is not automatically the best choice if it increases memory usage and inference cost without improving practical performance. Similarly, a very small model is not automatically the best choice if it cannot provide reliable field-test accuracy. Therefore, the final system requires a model that is both computationally efficient and robust. From this perspective, ResNet18 remains a strong candidate because it achieved the best offline performance, has a moderate model size compared to deeper CNNs and transformer-based models, and is more suitable for real-time deployment than very large architectures.

During the dataset extension stage, additional UAV-related controller signal data from the RFUAV dataset was also examined. The initial motivation was to make the system more general by considering not only drone model signals, but also different UAV-related RF communication signals. However, these controller signals were not directly included in the final downsampled 40 MHz training dataset. Some RFUAV recordings were originally sampled at 100 MHz, while the final project pipeline was organized around the 40 MHz processing structure. When the controller signals were downsampled, aliasing effects were observed in the generated STFT representations. In addition, the duty cycles of the controller signals were relatively low, which means that many STFT windows did not contain strong or clearly visible signal activity. As a result, directly mixing these downsampled controller signals with the drone model spectrogram dataset was not considered reliable.

This observation provides an important future work direction. A more reliable controller-signal classification model may be trained by preserving the original 100 MHz sampling rate and designing a dedicated preprocessing pipeline for these signals. Therefore, instead of forcing these signals into the current 40 MHz drone model dataset, they should be treated as a separate extension of the system.

Overall, the results show that the proposed STFT-based classification pipeline provides very high offline performance for both the 6-class and extended 9-class datasets. The 1024-point STFT representation was sufficient for preserving discriminative time-frequency information while reducing the computational cost of the GUI pipeline. The model comparison showed that larger models do not always improve performance and that lightweight models do not always provide reliable field-test behavior. Therefore, the final model should be selected by considering offline accuracy, model size, processing time, confidence stability, and GUI-integrated field-test reliability together.

6.1.4 Signal Parameter Extraction Results

The signal parameter extraction algorithm’s performance regarding bandwidth, center frequency, and SNR estimation was also tested. The results are provided in Table 13 in the format of (μ, σ) where they denote the error mean and standard deviation respectively.

Table 13: Error Rate for Detection Systems

Signal Type	Center Frequency (MHz)	Bandwidth (MHz)	SNR (dB)
Drone Signals	(0,0)	(0,0)	(0.1,0.1)
Controller Signals	(0,0.1)	(0,0)	(0.1,0.1)
Wi-Fi Signals	(0,0)	(0,0.3)	(0.2,0.2)
Bluetooth Signals	(0,0)	(0,0)	(0,0.1)

The effects of Doppler shift were also tested for with no significant effects on the accuracy of the obtained parameters. Theoretically, the frequency shift at 2.4 GHz from a drone moving orthogonally, at 100 m/s can be calculated as:

$$f' = f \left[1 + \frac{u \cos \theta}{c} \right] = 2.4505 \times 10^7 \left[1 + \frac{100}{3 \times 10^8} \right] = 2.4505 \times 10^7 + 8.168\bar{3} \quad (27)$$

$$\Delta f = \frac{40 \times 10^6}{4096} = 9765.625 Hz \gg 8.16 Hz \quad (28)$$

Hence, Doppler effects will not have any observable effect on the system.

6.2 Discussions and Lessons Learned

As a group, we were quickly able to propose a system that would achieve the desired performance. Hence, we were accurately able to identify the necessary components that needed to be developed. However, we failed to foresee the possible issues that arose during the development of these individual components. When using different datasets that we found online, their incompatibility made it difficult to obtain consistent results. However, after the arrival of the SDR and the drones, we were able to create our own cohesive dataset that featured rich variety of conditions, which again lead to significant acceleration of the progress being made.

We have observed that having the necessary theoretical background and experience that was required for the project was crucial, especially in areas like signal processing and neural networks. Throughout the project, we kept improving our theoretical foundations, learning about the literature, and gathering experience regarding the preferred tools, which again significantly accelerated our progress towards the end of the project.

Another lesson about neural network training is the importance of a balanced and cohesive dataset for training. At times, the contents of the dataset proved to be more influential

on model accuracy than the architecture itself. Possible contamination significantly hindered the performance of the FFT-Based model. On the other hand, the STFT-Based model’s accuracy suffered from the imbalance of recordings with or without Wi-Fi interference. If the model was not exposed to Mavic Air 2 recordings with interference, it could not accurately classify real-life Mavic Air 2 data with interference.

6.3 Future Directions

- Optimization, fine-tuning, comparative analysis and ablation studies for all DL models will continue. The classification model will be expanded with more drone classes.
- The multiple detection algorithm can be optimised to support multiple time frames in parallel to process the FFT-frames in batches to accelerate the process.
- Alternative approaches (such as SegNet) will continue to be developed and possibly integrated to the GUI.
- DroneID detection will be developed as an independent framework or integrated into the system.
- A more reliable controller-signal classification model may be trained by preserving the original 100 MHz sampling rate and designing a dedicated preprocessing pipeline for these signals. Therefore, instead of forcing these signals into the current 40 MHz drone model dataset, they should be treated as a separate extension of the system.
- The CRNN-based classification pipeline will be completed by training and evaluating CRNN architectures on both modalities. This approach is expected to better capture the temporal structure of drone control bursts and packet cadence.

7 Detailed Equipment List

The main equipment used in this project are listed in Table 14, namely the SDR and the drone models.

Table 14: Detailed Equipment List (*Currency Conversion According to 8 May. 2026 Rates)

Requirement	Product Model	QTY	Unit Cost (TRY/USD)	Provided By
SDR	USRP B200mini [32]	1	68,286/1506*	Meteksan
Drone	DJI Mavic Air [33]	1	5,832/129	
Drone	DJI Mini 3 [34]	1	42,341/934	
Drone	DJI Mini 4 Pro [35]	1	56,667/1250	

Total Cost = 130,785, 3,819

References

- [1] A. Solodov, A. Williams, S. Al Hanaei, and B. Goddard, “Analyzing the threat of unmanned aerial vehicles (uav) to nuclear facilities,” *Security Journal*, vol. 31, pp. 305–324, 2018.
- [2] B. Nassi, A. Shabtai, R. Masuoka, and Y. Elovici, “Sok - security and privacy in the age of drones: Threats, challenges, solution mechanisms, and scientific gaps,” 2019. [Online]. Available: <https://arxiv.org/abs/1903.05155>
- [3] A. Alajami, A. Vulpe, and O. Fratu, “Uavs for wi-fi receiver mapping and packet sniffing with antenna radiation pattern diversity,” *Wireless Personal Communications*, vol. 92, 01 2017.
- [4] B. Kaplan, Kahraman, A. Görçin, H. A. Çırpan, and A. R. Ekti, “Measurement based fhss-type drone controller detection at 2.4ghz: An stft approach,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–6.
- [5] C. Xu, B. Chen, Y. Liu, F. He, and H. Song, “Rf fingerprint measurement for detecting multiple amateur drones based on stft and feature reduction,” in *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*, 2020, pp. 4G1–1–4G1–7.
- [6] T. D. Gaikwad and S. V. Dhavale, “Rf based enhanced drone detection and classification using short time fourier transform,” in *2024 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI)*, 2024, pp. 1–6.
- [7] R. Shi *et al.*, “Rfuav: A benchmark dataset for unmanned aerial vehicle detection and identification,” *arXiv preprint arXiv:2503.09033*, 2025.
- [8] R. Kılıç, N. Kumbasar, E. A. Oral, and I. Y. Ozbek, “Drone classification using rf signal based spectral features,” *Engineering Science and Technology, an International Journal*, vol. 28, p. 101028, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2215098621001403>
- [9] K. N. Inani, K. Sangwan, and Dhiraj, “Machine learning based framework for drone detection and identification using rf signals,” in *2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 2023, pp. 1–8.
- [10] M. Almasri, “Deep learning for rf-based drone detection and identification using welch’s method,” in *Proceedings of the 10th International Conference on Data Science, Technology and Applications (DATA 2021)*. SCITEPRESS, 2021, pp. 208–214.
- [11] Aartos, “Systems and versions,” <https://drone-detection-system.com/aartos-dds/systems-versions/>, (Accessed Dec. 27, 2025).
- [12] —, “Software,” <https://drone-detection-system.com/aartos-dds/software-integration/>, (Accessed Dec. 27, 2025).
- [13] AeroDefense, “Airwarden essentials,” <https://aerodefense.tech/drone-remote-id-drone-detection-airwarden/>, (Accessed Dec. 27, 2025).

- [14] B. Sazdić-Jotić, I. Pokrajac, J. Bajčetić, B. Bondžulić, and D. Obradović, “Single and multiple drones detection and identification using rf based deep learning algorithm,” *Expert Systems with Applications*, vol. 187, p. 115928, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417421012823>
- [15] M. F. Al-Sa’d, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, “Rf-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database,” *Future Generation Computer Systems*, vol. 100, pp. 86–97, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X18330760>
- [16] Y. Jararweh, M. Daraghme, and A. Agarwal, “Improved rf-based drone detection via advanced feature extraction and sigmoid calibration,” in *NAECON 2024 - IEEE National Aerospace and Electronics Conference*, 2024, pp. 237–240.
- [17] D. W. Kleinbeck, “Unmanned vehicle recognition and threat management,” US Patent US11 521 498B2, Dec. 6, 2022. [Online]. Available: <https://patents.google.com/patent/US11521498B2/en>
- [18] S. Hamminga, B. Portegijs, and H. J. L. Westra, “Detection and classification of unmanned aerial vehicles,” European Patent EP3 715 901B1, Jul. 31, 2024. [Online]. Available: <https://patents.google.com/patent/EP3715901B1/en>
- [19] “Dronerf dataset: Rf-based drone detection, classification and identification,” *ResearchGate*, 2019, accessed: Nov. 1, 2025. [Online]. Available: <https://www.researchgate.net/publication/335419489>
- [20] R. Shi, “Rfuav: A radio frequency dataset for uav detection and identification,” 2025. [Online]. Available: <https://huggingface.co/datasets/kitofrank/RFUAV>
- [21] “GNU Radio: The free and open software radio ecosystem,” <https://www.gnuradio.org>, 2024, accessed: 2025.
- [22] Tensorflow, “Tensorflow,” <https://www.tensorflow.org/>, (Accessed Dec. 23, 2025).
- [23] Pytorch, “Pytorch,” <https://pytorch.org/>, (Accessed Dec. 23, 2025).
- [24] The MathWorks, Inc., “Matlab,” <https://www.mathworks.com/products/matlab.html>, accessed 2025.
- [25] “Pyqt documentation,” <https://www.riverbankcomputing.com/software/pyqt/>, accessed 2025.
- [26] F. Auger and P. Flandrin, “Improving the readability of time-frequency and time-scale representations by the reassignment method,” *IEEE Transactions on Signal Processing*, vol. 43, no. 5, pp. 1068–1089, 1995.
- [27] D. Gabor, “Theory of communication,” *Journal of the Institution of Electrical Engineers*, vol. 93, pp. 429–457, 1946.
- [28] F. C. Akyon, Y. K. Alp, G. Gok, and O. Arikan, “Classification of intra-pulse modulation of radar signals by feature fusion based convolutional neural networks,” in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 2290–2294.
- [29] R. Shi, X. Yu, S. Wang, Y. Zhang, L. Xu, P. Pan, and C. Ma, “Rfuav: A benchmark dataset for unmanned aerial vehicle detection and identification,” *arXiv preprint arXiv:2503.09033*, 2025.
- [30] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [31] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, “Ablation studies in artificial neural networks,” *arXiv preprint arXiv:1901.08644*, 2019.
- [32] Ettus, “Usrc x310,” <https://www.ettus.com/all-products/x310-kit/>, (Accessed Dec. 26, 2025).

- [33] Drone, “Dji mavic air,” <https://drone.net.tr/drone-modelleri/profesyonel-drone/dji-mavic-air.html>, (Accessed Dec. 26, 2025).
- [34] DJI Store Türkiye, “Dji mini 3 fly more combo (dji rc) - 128sd kart hediye,” <https://www.djistoreturkiye.com/dji-mini-3-fly-more-combo--dji-rc---128sd-kart-hediye-6941565979667>, (Accessed Dec. 26, 2025).
- [35] Teknosa, “Dji mini 4 pro fly more rc 2 combo plus drone multikopter,” <https://www.teknosa.com/dji-mini-4-pro-fly-more-rc-2-combo-plus-drone-multikopter-p-789630285>, (Accessed Dec. 26, 2025).
- [36] International Electrotechnical Commission, *IEC 61010-1: Safety requirements for electrical equipment for measurement, control, and laboratory use*, IEC Std., 2010.
- [37] European Telecommunications Standards Institute, *ETSI EN 301 489: Electromagnetic compatibility (EMC) standard for radio equipment*, ETSI Std., 2019.
- [38] IEEE Standards Association, *IEEE C95.1: Safety Levels with Respect to Human Exposure to Electric, Magnetic, and Electromagnetic Fields*, IEEE Std., 2019.
- [39] International Commission on Non-Ionizing Radiation Protection, “Guidelines for limiting exposure to electromagnetic fields,” *Health Physics*, vol. 118, no. 5, pp. 483–524, 2020.
- [40] European Parliament and Council of the European Union, “Directive 2011/65/eu on the restriction of the use of certain hazardous substances in electrical and electronic equipment (rohs),” 2011, official Journal of the European Union.
- [41] IEEE Standards Association, *IEEE 802.3: Ethernet Standard*, IEEE Std., 2018.
- [42] USB Implementers Forum, *Universal Serial Bus Specification*, USB-IF Std., 2019.
- [43] IEEE Standards Association, *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., 2020.
- [44] Bluetooth SIG, *Bluetooth Core Specification*, Bluetooth SIG Std., 2021.
- [45] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: <https://api.semanticscholar.org/CorpusID:14124313>
- [46] M. Kadhim, H. Al-Sahaf, B. Xue, and M. Zhang, “Training-free neural architecture search via total variation penalty,” 2024. [Online]. Available: <https://arxiv.org/abs/2409.05569>
- [47] S. Kato and K. Hotta, “Adaptive t-vmf dice loss for multi-class medical image segmentation,” 2022. [Online]. Available: <https://arxiv.org/abs/2207.07842>
- [48] S. Bomble, “Understanding sparse categorical cross-entropy and binary cross-entropy,” Medium, Feb. 2025, accessed: 2026-05-05. [Online]. Available: <https://medium.com/@shivamvbomble/understanding-sparse-categorical-cross-entropy-and-binary-cross-entropy-d37aa33aed8e>

1 Detailed Gantt Chart

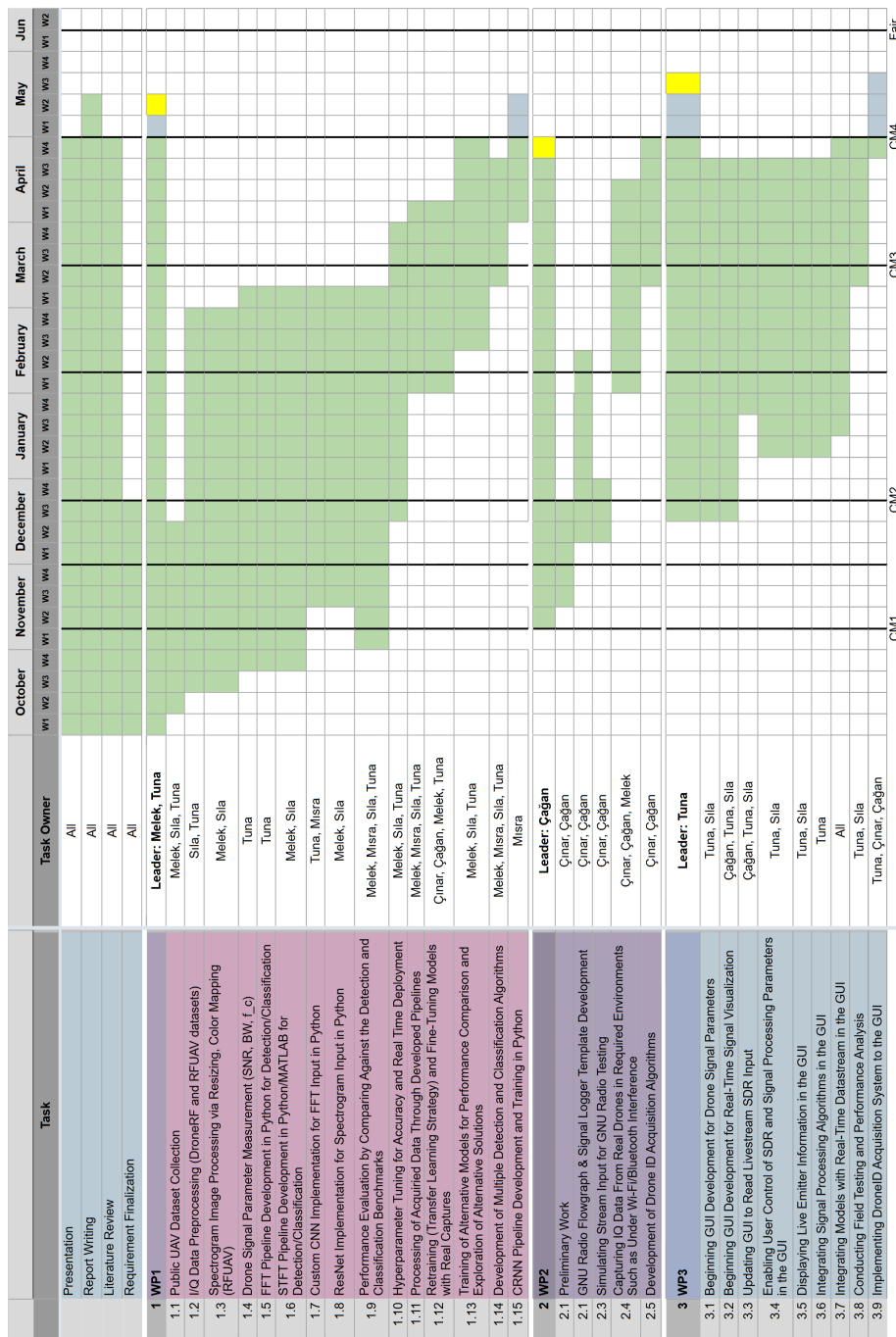


Table 15: Detailed Gantt Chart

2 Non-Functional Requirements, Standards

This appendix shows the explanations regarding the standards referenced in the non-functional requirements.

Standards Explanation: IEC 61010-1 defines general safety requirements for electrical equipment used in measurement, control, and laboratory environments [36]. ETSI EN 301 489 specifies electromagnetic compatibility requirements to ensure that radio equipment operates without causing or suffering from harmful interference [37]. IEEE C95.1 and the IC-NIRP 2020 guidelines define recommended limits for human exposure to electric, magnetic, and electromagnetic fields to ensure health and safety during system operation [38, 39]. The RoHS Directive 2011/65/EU restricts the use of hazardous substances in electrical and electronic equipment to reduce environmental impact [40]. Communication and data transfer standards such as Ethernet and USB are included due to their use in SDR hardware connectivity, while Wi-Fi and Bluetooth standards are referenced to characterize common interfering communication signals considered during detection and classification [41–44].

3 Detailed System Architecture

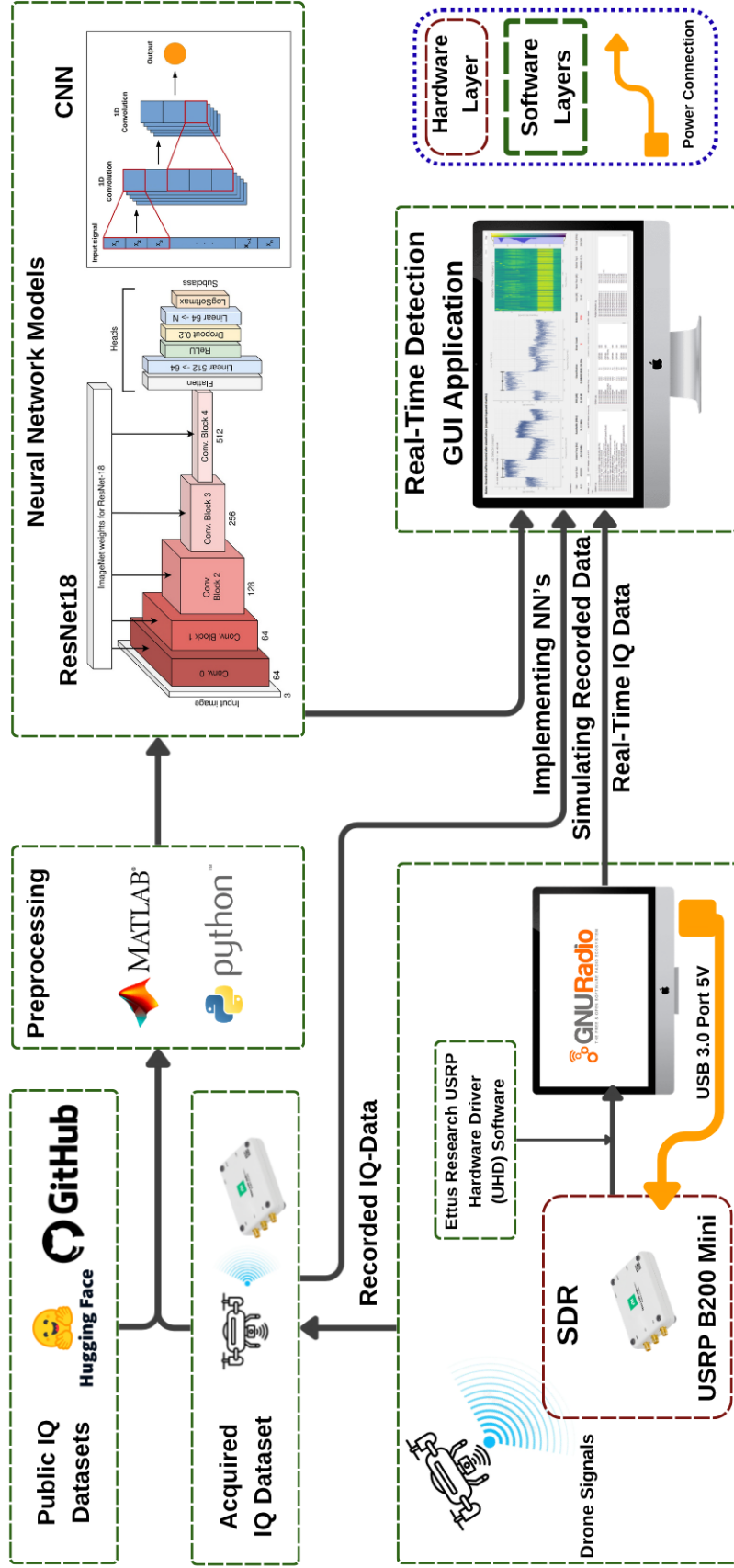


Figure 16: Detailed system architecture diagram highlighting the hardware and software components and their interconnections.

4 Reassigned STFT Generation and VGG19 Model

This appendix features the progress relating to the reassigned-STFT-based VGG19 model. A new dataset consisting of reassigned spectrograms was generated. The STFT was taken with the Hanning window with an FFT size of 1024 and 75% overlap, and saved in the format of 512x256 px PNG files. A model utilizing the VGG-19 architecture [45] has been trained on this dataset and its performance has been evaluated.

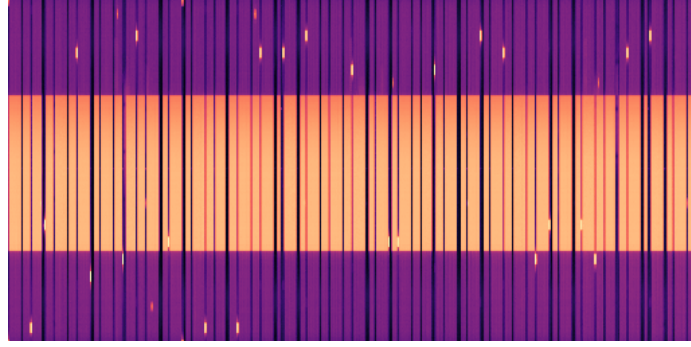


Figure 17: Example Reassigned Spectrogram (DJI MAVIC AIR 2)

An example reassigned spectrogram can be seen in Fig.17. Drone communication signals, both the FHSS and the map transmission signal, are quite stationary by themselves and reassignment does not alter the fundamental structure significantly.

The trained VGG19 model has 100% accuracy on the dataset and the confusion matrix is shown in Fig.18, yet its large size (approx. 545 MB) and consequently its long feed-forward time of 180 ms on average in a GPU makes it unfeasible to be deployed in a real-time classification setting. The process of reassignment and image generation also requires a considerable amount of time (approx. 4.3 sec) so the latency from the method itself also makes it unlikely to be used with another model.

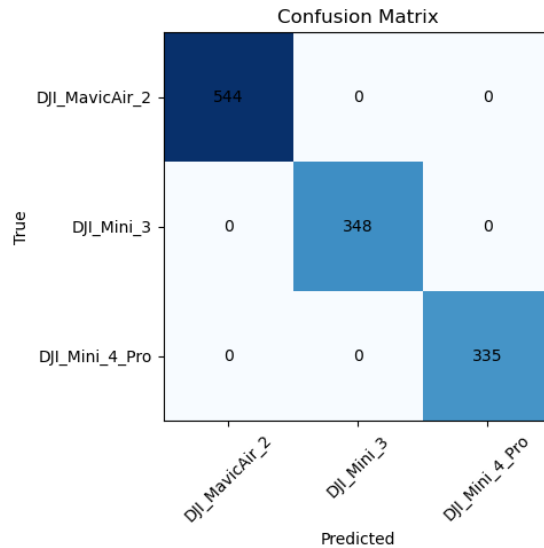


Figure 18: Confusion Matrix for VGG19 Model on Reassigned Spectrogram Dataset

5 FFT-Based Custom Segmentation Network - SegNet

This appendix features the progress related to the SegNet model. Another FFT-Based Convolutional Neural Network was developed with the intention of performing multiple detection and center frequency and bandwidth extraction solely through this new model. The labels for this model were generated using a modified version of the multiple detection algorithm. The designed network aims to classify each FFT-bin with its corresponding emitter type. Hence, the network has an input of size $[\text{fft_len}, 1]$ and output of size $[\text{fft_len}, 3]$. Where each bin's probabilities for each type are outputted. Then the argmax function was used to determine the type of each bin. Custom loss functions were implemented as a mixture of total-variation (TV) penalties [46], dice roll loss [47], and sparse categorical cross-entropy [48]. Example inputs and outputs for this new network are shown in Fig. 19.

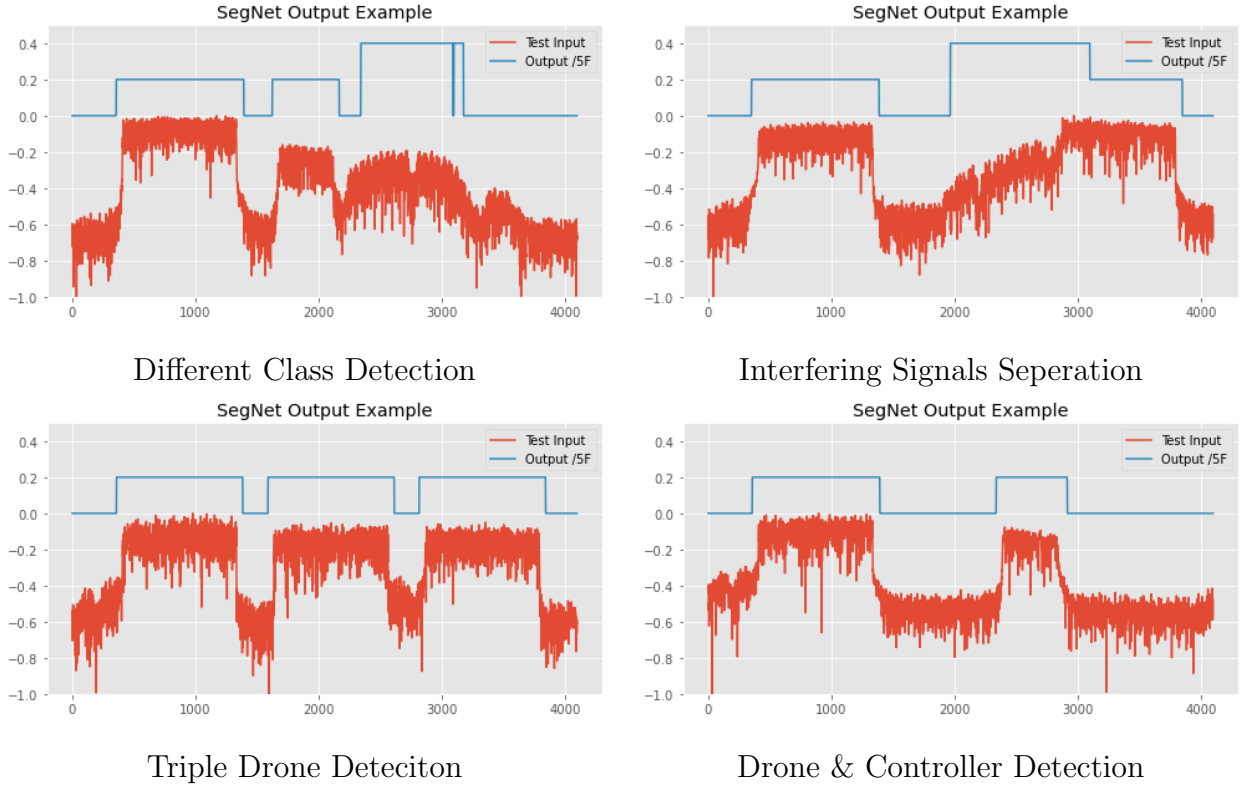


Figure 19: SegNet Inputs, Labels, and Outputs

The outputs of the network are of the form: $o[n] = \sum_i b_i[k]T(i)$ where, $T(\text{Noise}) = 0$, $T(\text{Drone/Controller}) = 1$, $T(\text{Wi-Fi/Bluetooth}) = 2$. This output can then directly be used to calculate the bandwidth and center frequency using a similar approach to the signal parameter extraction algorithm (see section 5.2.2.1.2). This model was mostly accurate at detecting and classifying signals. The main drawbacks of the model were accurately estimating Wi-Fi signals' bandwidths which reflects poorly on the binwise Wi-Fi classification accuracy. Spectrum-wide classification accuracy was much higher as both are shown in Appendix 6. The performance significantly suffered at SNRs where the multiple detection algorithm could still perform. The critical SNR was found to be around 10dB.

6 FFT-Based Network Accuracy Results

The confusion matrices for the FFT-based classifier CNN and the FFT-based SegNet are provided in this appendix along with the accuracy performance comparison for different sized FFT-Based classifier CNN.

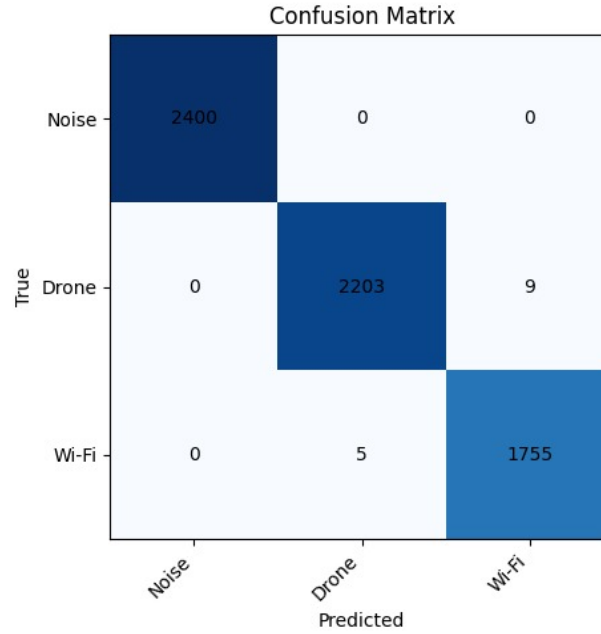


Figure 20: Dataset Augmented FFT-Based CNN Confusion Matrix

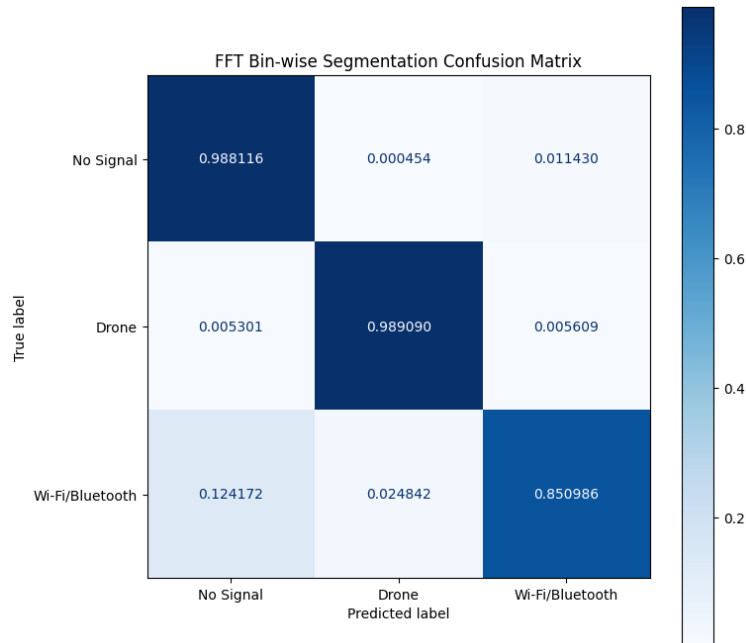


Figure 21: Bin-Wise Normalised Confusion Matrix for SegNet

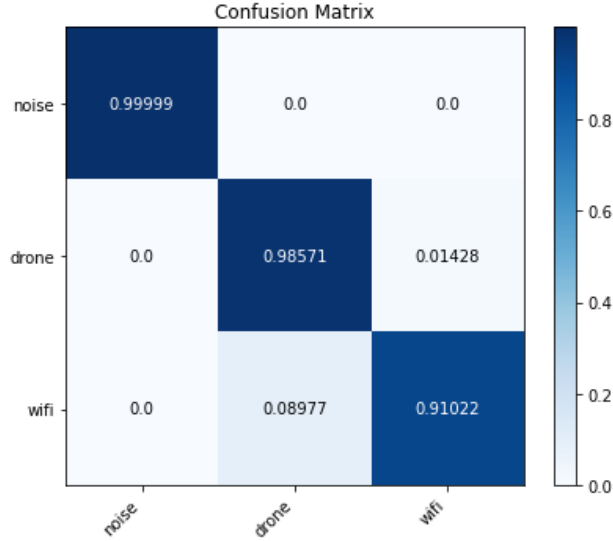


Figure 22: Classification-Wise Normalised Confusion Matrix for SegNet

In Table 16, different sized FFT-Based model's performances are evaluated based on offline and real-life accuracy, feed forward times, and biases. Overperception bias indicates that the model is more likely to give false positives compared to false negatives and underperception bias indicates that the model is more likely to give false negatives compared to false positives. In this project false positives are much more costly as a single false negative can be immediately compensated since drone signals occur at multiple frames at once. Dataset C is the 12th generation of dataset in development as C represents the generation in hexadecimal. C2 is the C dataset downsampled by 2, and D dataset is the 13th generation dataset.

Model Name	Input Size	Conv 1	Conv 2	Dense 1	Dense 2	Offline Acc. (%)	GUI Acc. (%)	File Size	Conf. Mat. Time	Bias	Dataset
cnn_gen6_exp_fp_v3.keras	4096	64	128	96	8	99.20	99.20	724 kB	9343 ms	No Bias	C
main_halved.keras	4096	32	64	48	8	99.03	93.24	242 kB	4113 ms	No Bias	C
main_third.keras	4096	22	44	33	8	99.17	90.10	156 kB	3740 ms	Underperception Bias	C
downsampled_full.keras	2048	64	128	96	8	99.37	87.13	724 kB	3740 ms	Overperception Bias	C2
downsampled_halved.keras	2048	32	64	48	8	99.31	83.10	242 kB	2718 ms	No Bias	C2
downsampled_third.keras	2048	22	44	33	8	99.27	82.00	156 kB	2420 ms	Underperception Bias	C2
cnn_gen6_fp_v3_aug.keras	4096	64	128	96	8	99.80	99.91	724 kB	9398 ms	Underperception Bias	D
augmented_halved.keras	4096	32	64	48	8	99.03	93.24	242 kB	4007 ms	No Bias	D
augmented_third.keras	4096	22	44	33	8	99.17	90.10	156 kB	3841 ms	Underperception Bias	D

Table 16: Comparison of CNN model architectures, accuracy results, file sizes, FFT confusion matrix times, bias observations, and datasets.

7 Confusion Matrices of Trained Classification Models

This appendix presents the confusion matrices of the trained classification models. These matrices provide a class-based view of the prediction performance and help identify possible confusions between drone classes.

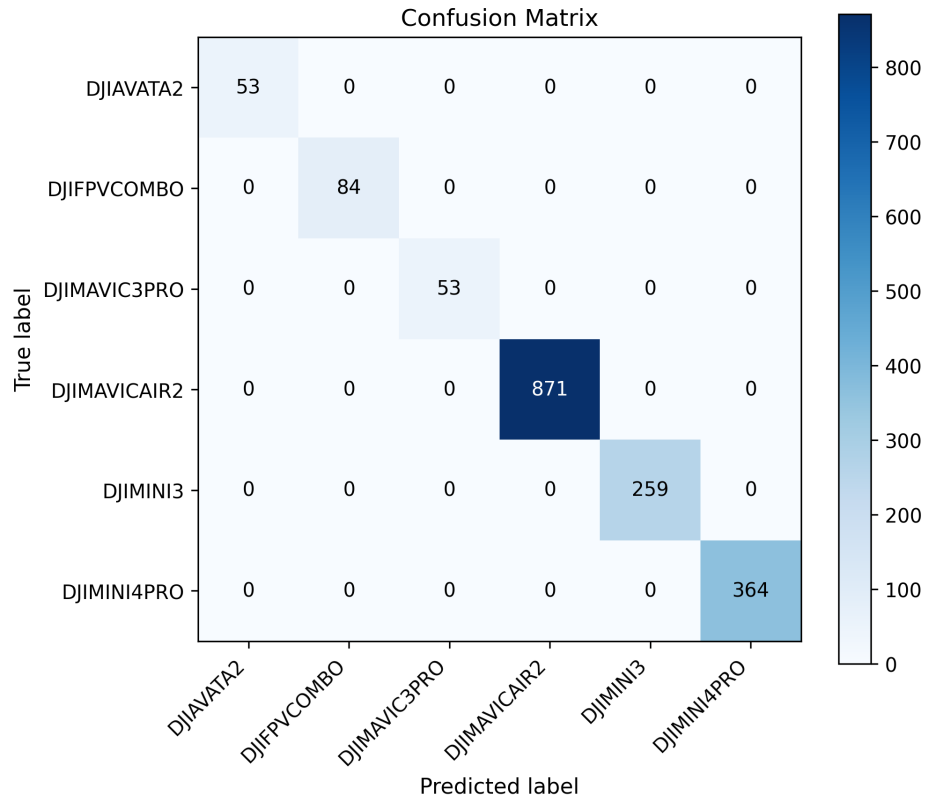


Figure 23: Confusion matrix of the ResNet18 model for the 6-class dataset.

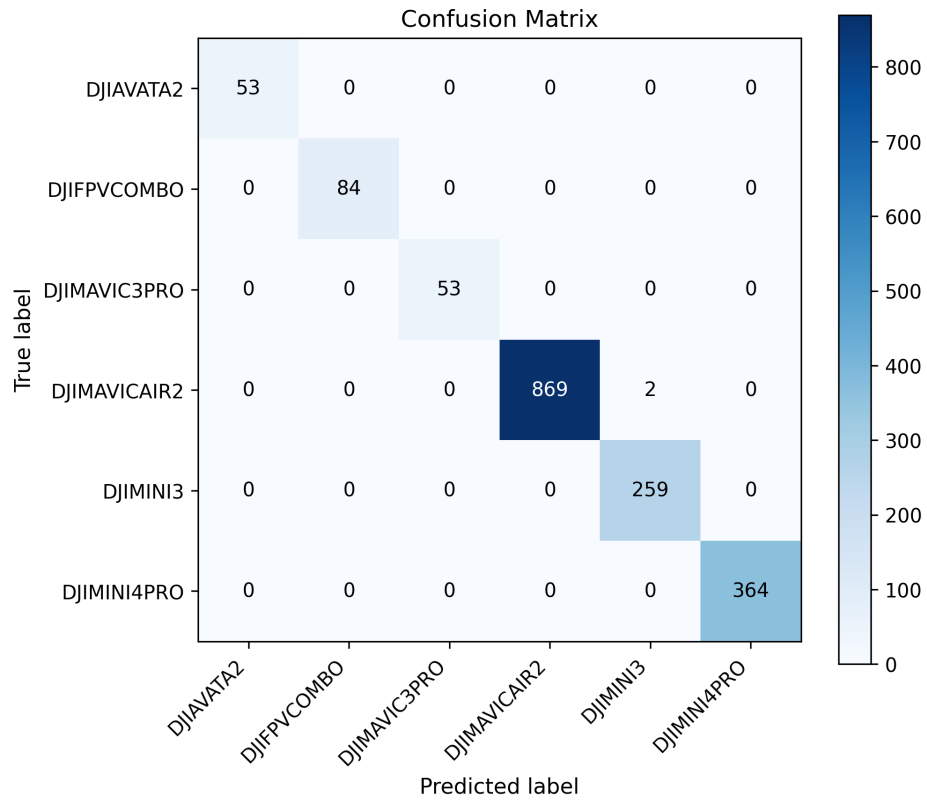


Figure 24: Confusion matrix of the ResNet34 model for the 6-class dataset.

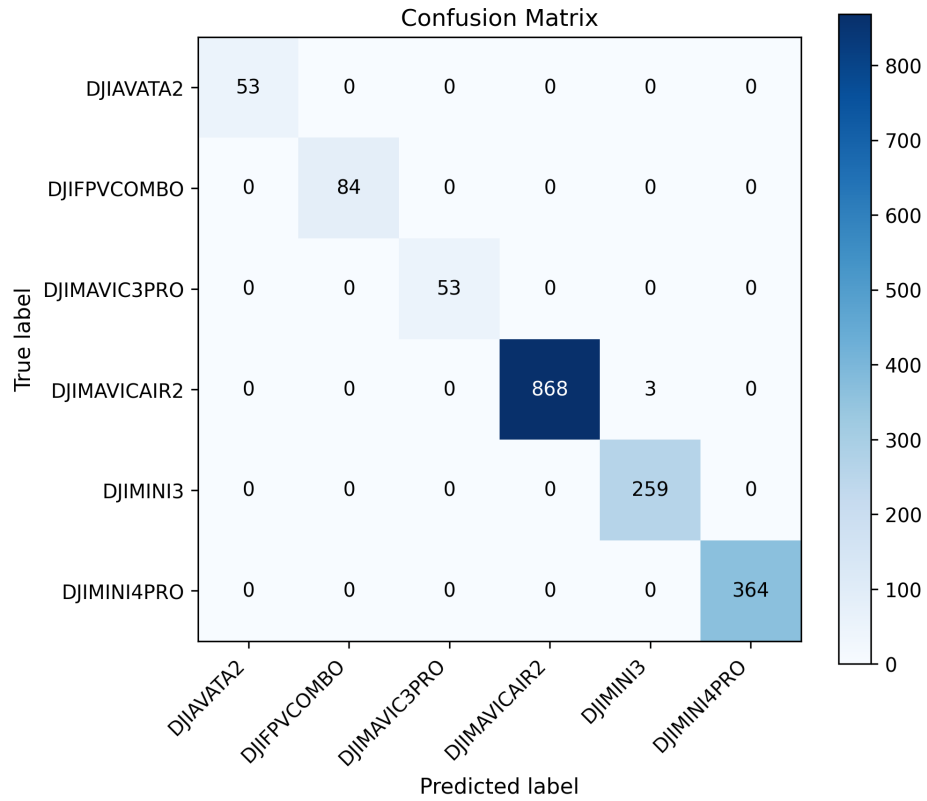


Figure 25: Confusion matrix of the ResNet50 model for the 6-class dataset.

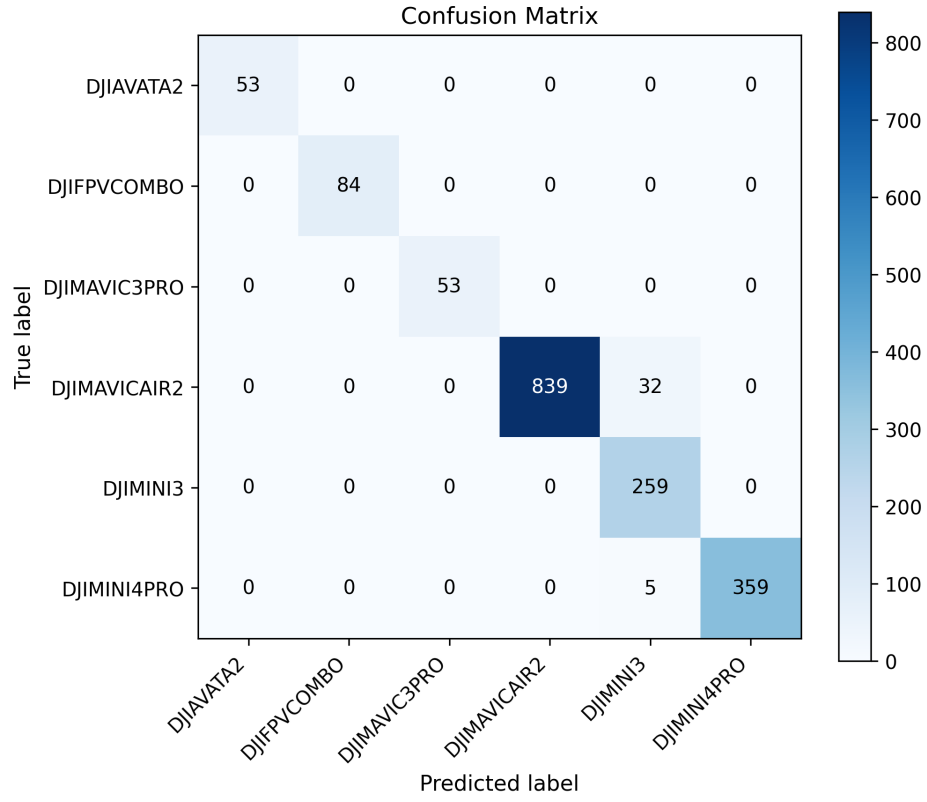


Figure 26: Confusion matrix of the ResNet101 model for the 6-class dataset.

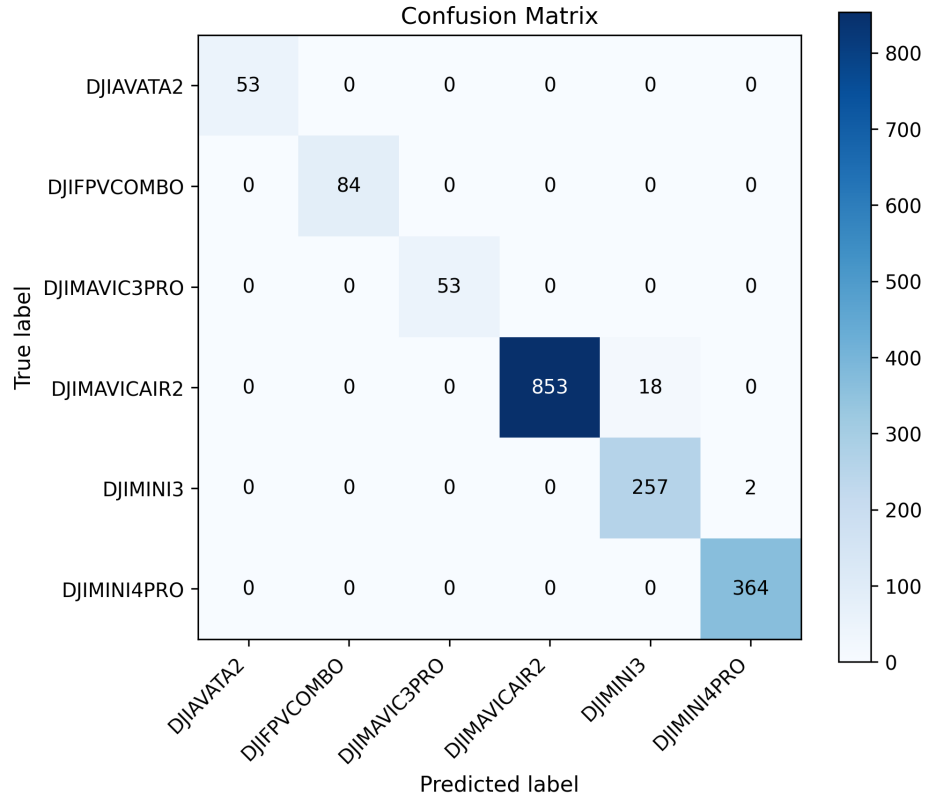


Figure 27: Confusion matrix of the SwinV2-Tiny Transformer model for the 6-class dataset.

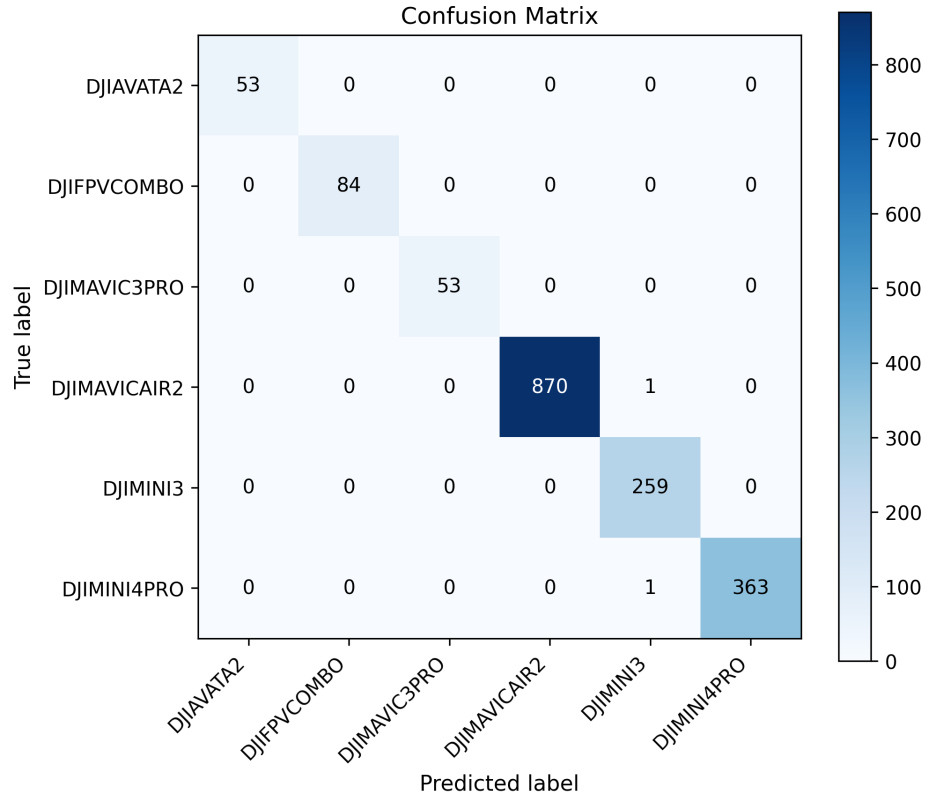


Figure 28: Confusion matrix of the ViT-B/16 model for the 6-class dataset.

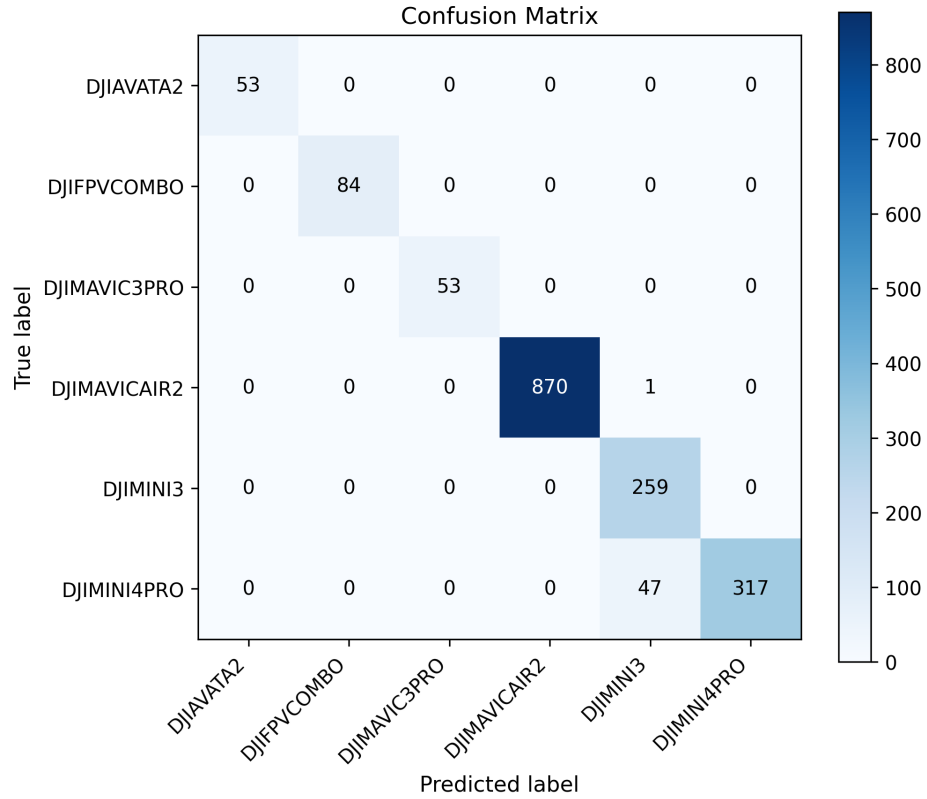


Figure 29: Confusion matrix of the ViT-L/16 model for the 6-class dataset.

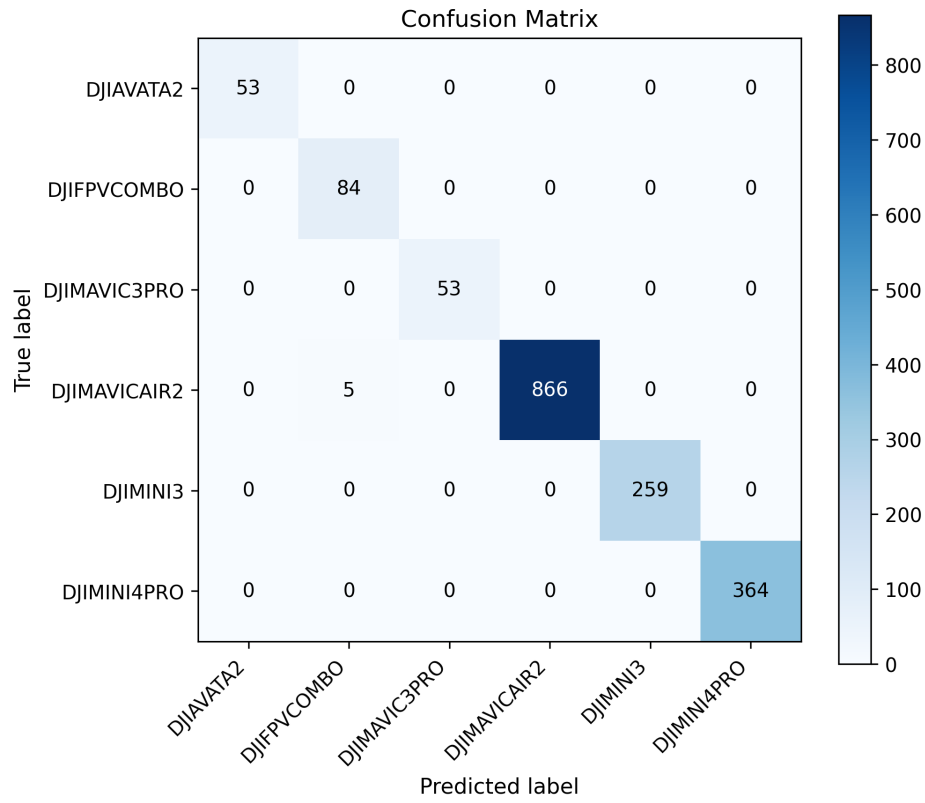


Figure 30: Confusion matrix of the MobileNetV3-Small model for the 6-class dataset.

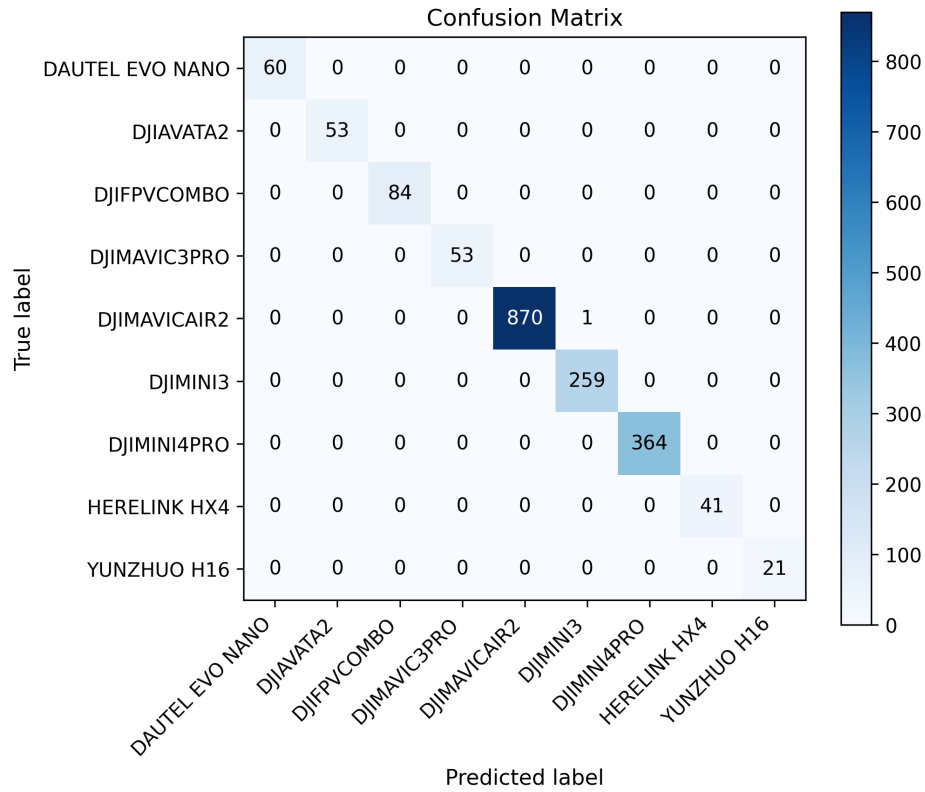


Figure 31: Confusion matrix of the ResNet18 model for the extended 9-class dataset.

8 FFT-Based GPU Training Results

This appendix shows the results of the FFT-based CNN training pipeline after migration to the GPU-accelerated Colab environment, along with the comparison of model performance with and without the Gaussian noise augmentation layer.

Fig. 32 shows the confusion matrix of the FFT-based CNN trained with the Gaussian noise augmentation layer enabled. The model achieves an overall accuracy of 98.60% with a macro F1-score of 98.56%.

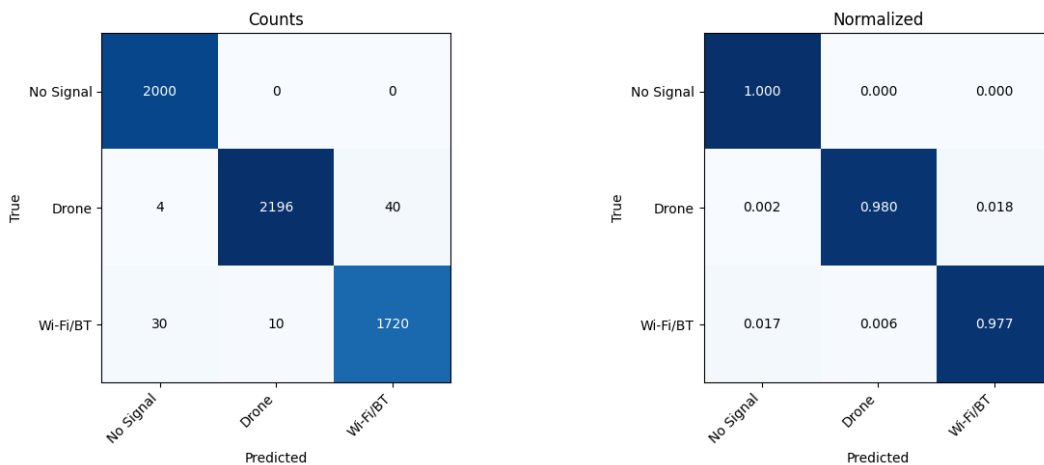


Figure 32: Gaussian Model Confusion Matrices

Fig. 33 shows training and validation loss/accuracy curves of the FFT-based CNN trained with the Gaussian noise augmentation layer enabled. The curves show slightly more stochastic behavior during training due to the injected noise.

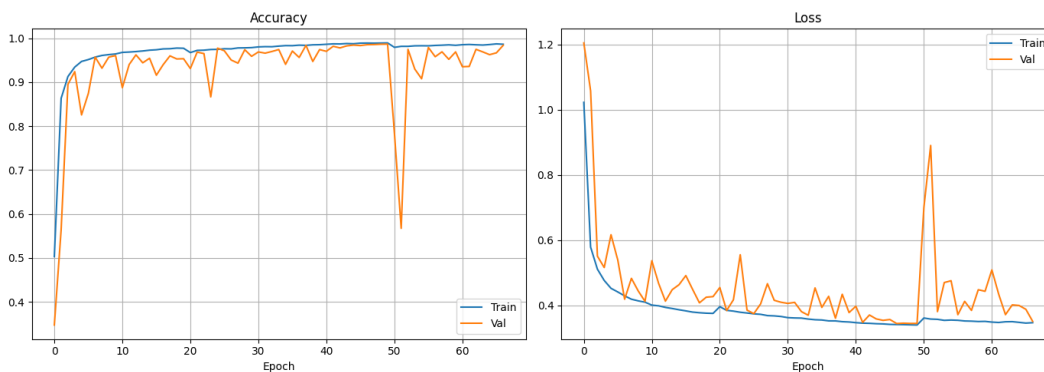


Figure 33: Gaussian Model Loss

Fig. 34 shows the confusion matrix of the FFT-based CNN trained without the Gaussian noise augmentation layer. The model achieves an overall accuracy of 98.37%.

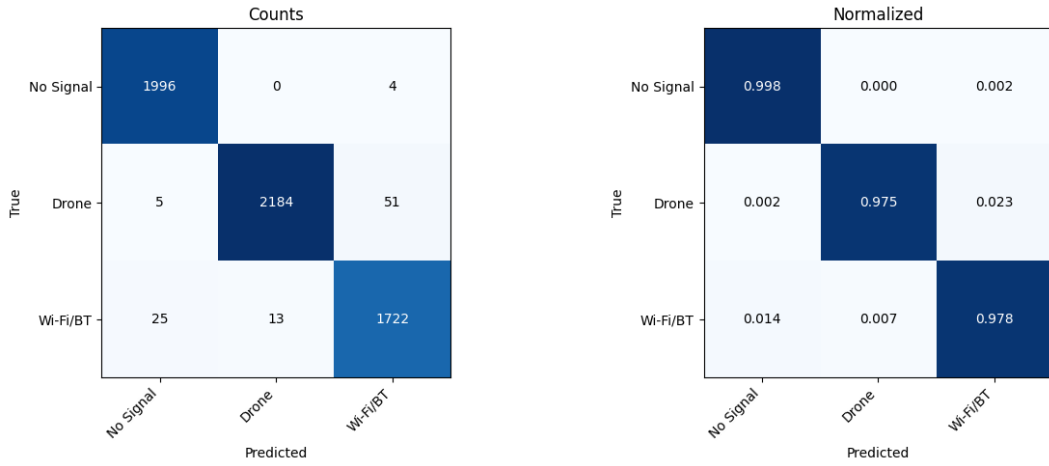


Figure 34: Gaussian-Free Model Confusion Matrices

Fig.35 training and validation loss/accuracy curves of the FFT-based CNN trained without the Gaussian noise augmentation layer. Convergence is faster and validation curves are smoother compared to the model trained with Gaussian noise.

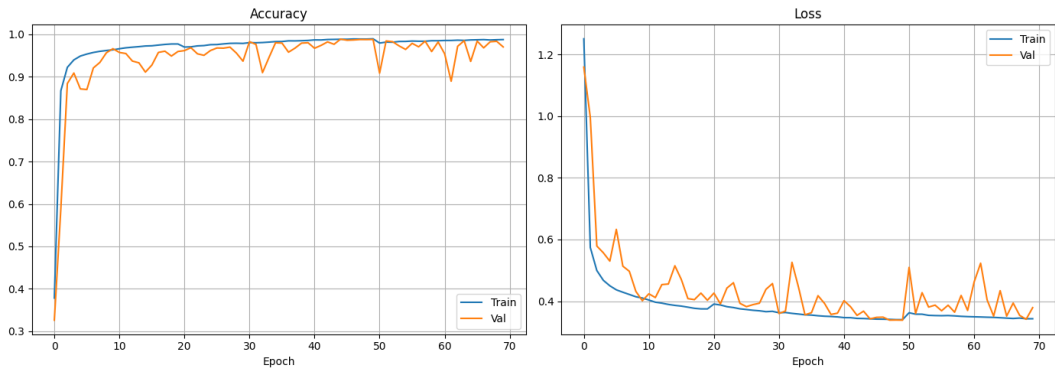


Figure 35: Gaussian-Free Model Loss

Metric	Without Gaussian Noise	With Gaussian Noise
Test Accuracy	98.37 %	98.60 %
Test Loss	0.3371	0.3412
Macro Precision	98.28 %	98.53 %
Macro Recall	98.38 %	98.59 %
Macro F1-score	98.32 %	98.56 %
Drone Recall	97.50 %	98.04 %
Wi-Fi/BT Recall	97.84 %	97.73 %
Inference Latency	19.5 ms	19.5 ms
Model Parameters	105,219	105,219
TFLite Size	116 kB	116 kB

Table 17: Performance comparison of the FFT-based CNN trained on the GPU-accelerated Colab pipeline with and without the Gaussian noise augmentation layer. Both models share identical preprocessing, mixed-precision training, warmup + cosine annealing schedule, label smoothing, and test sets.

9 Ablation Study Results

This appendix presents Table 18 which summarizes the results of the conducted ablation study on four different ResNet18-based model configurations. Each configuration was evaluated independently in order to examine the effect of the Squeeze-and Excitation (SE) attention mechanism and the redesigned classifier head on classification performance and computational efficiency. In addition to classification metrics, forward-pass inference latency, parameter count, confidence score, and model size are also reported to provide a broader comparison between the evaluated architectures.

The redesigned custom classifier head consists of a sequential architecture containing Batch Normalization followed by multiple fully connected linear layers, ReLU activation functions, and a Dropout layer to improve feature learning capability, stabilize training, and reduce overfitting.

The *layer_4c* fine-tuning strategy was employed for each model variant which refers to freezing the earlier layers of ResNet18 while allowing only the final residual block (layer 4) and the fully connected classifier layers to be trainable.

Table 18: Ablation Study Results of Different ResNet18-Based Architectures

Model Variant	Accuracy	F1 Score	Latency (s)	Model Size (MB)
Baseline	1.0000	1.0000	0.4487	42.72
SE Only	1.0000	1.0000	0.4376	42.85
Custom FC Only	0.9919	0.9946	0.4438	43.35
Full Proposed Model	1.0000	1.0000	0.4347	43.48

10 CRNN Preprocessing Pipeline

This appendix presents the concepts for the CRNN-based pipeline. The pipeline in Fig. 36 shows raw IQ recordings are segmented into 4096-point frames, normalized using per-frame z-score, and assembled into file-boundary-aware sequences of 32 consecutive frames (stride 8) to prevent data leakage across recordings. The pipeline in Fig. 37 shows 30 ms IQ segments are converted into (128×1024) float16 spectrograms using 2048-point FFT with 512-sample hop and 75% segment overlap. A streaming variant writes spectrograms directly to memory-mapped disk arrays, enabling processing of ~ 60 GB datasets within Colab's 12 GB RAM limit. Fig. 38 shows the concept of how a CRNN model might be implemented. A TimeDistributed CNN encoder extracts features from each time step, and a stacked Bidirectional LSTM consumes the resulting sequence of embeddings, making temporal patterns such as drone control bursts and packet cadence directly learnable.

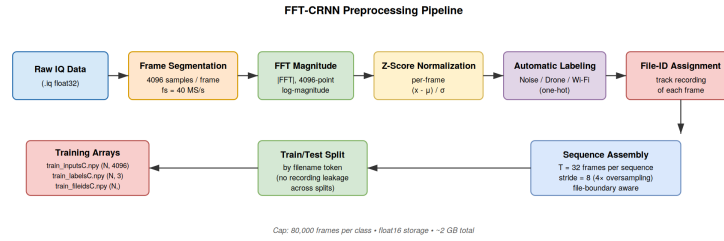


Figure 36: Block Diagram of the FFT-CRNN Preprocessing Pipeline

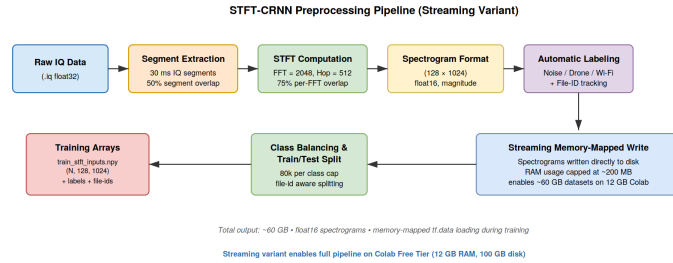


Figure 37: Block Diagram of the STFT-CRNN Preprocessing Pipeline

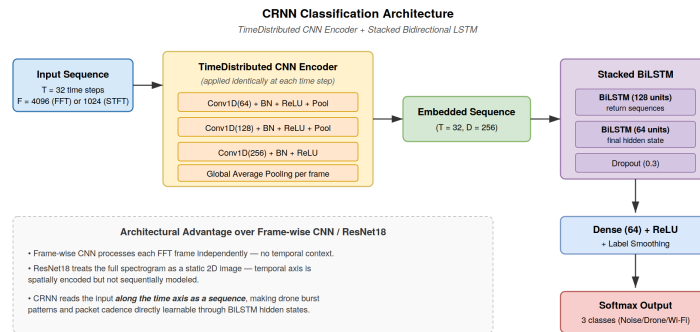


Figure 38: Conceptual architecture of the CRNN classification model

11 Drone ID Telemetry GNU Radio Block Diagram

This appendix has the block diagram used for the Drone ID telemetry process.

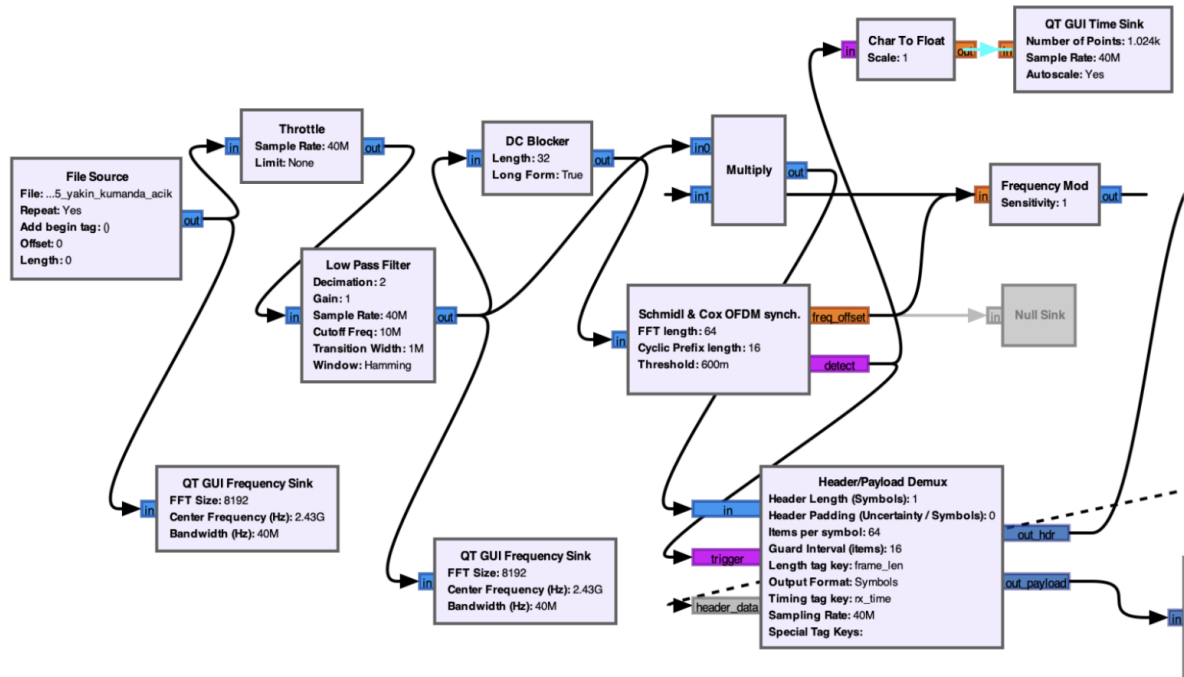


Figure 39: GNU Radio Block Diagram for Drone ID, Part 1/2

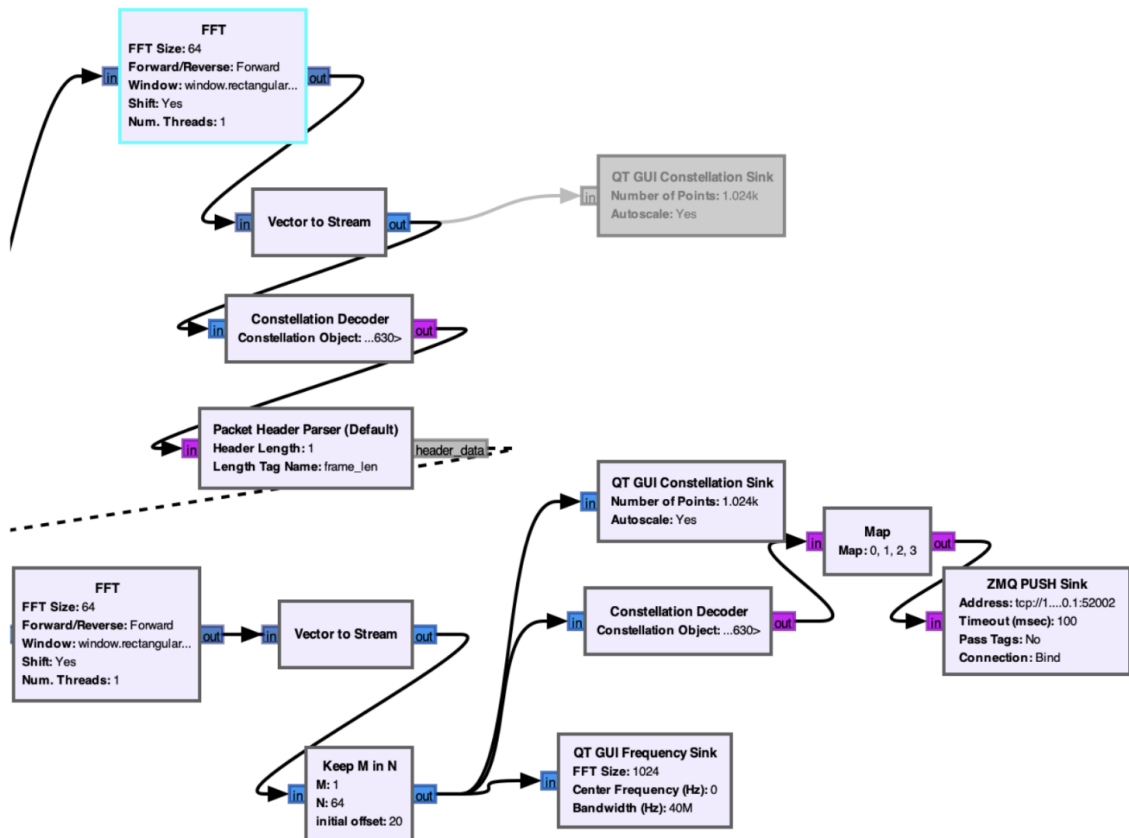


Figure 40: GNU Radio Block Diagram for Drone ID, Part 2/2

12 Graphical User Interface Pictures

This appendix features the images relating the GUI are provided in this appendix, such as the different color themes, GUI of CM3, and the detailed GUI.



Figure 41: GUI, Theme: Default Dart

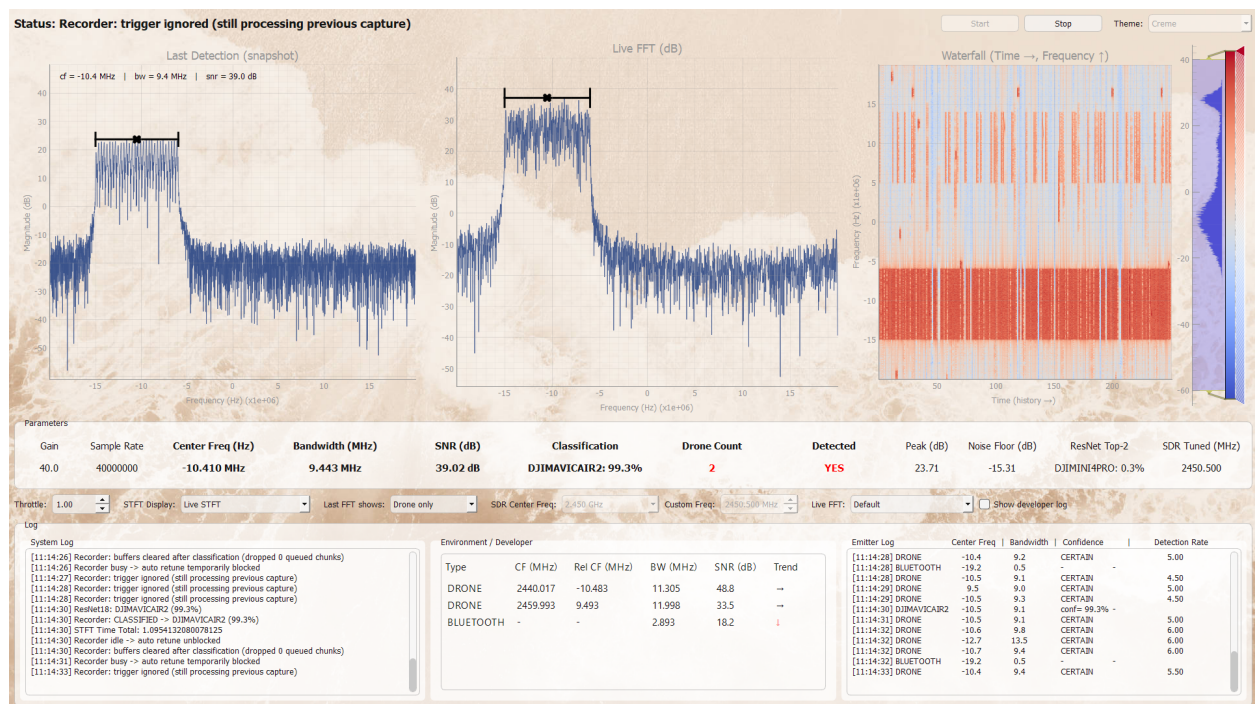


Figure 42: GUI, Theme: Creme

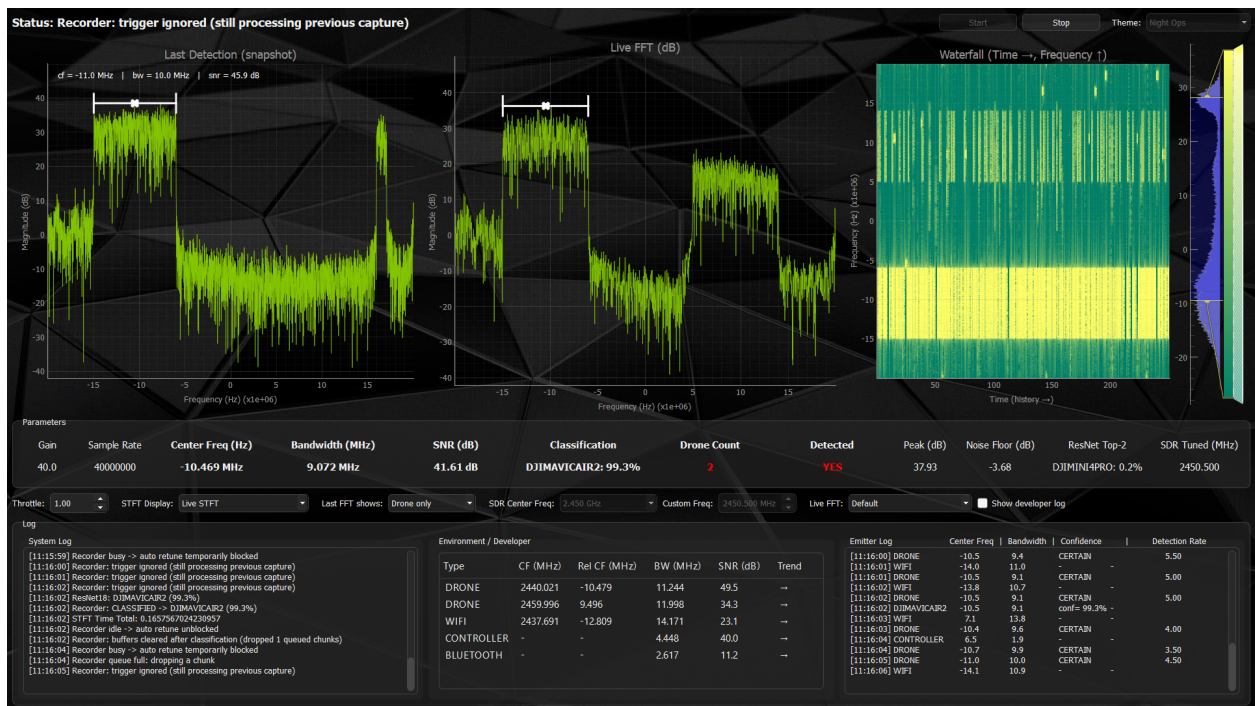
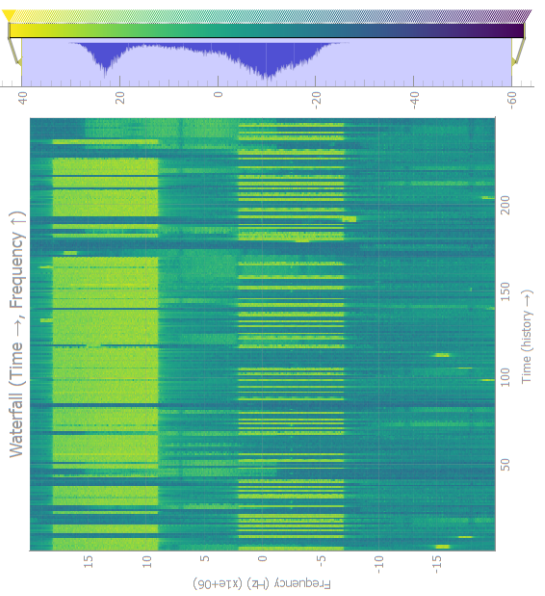
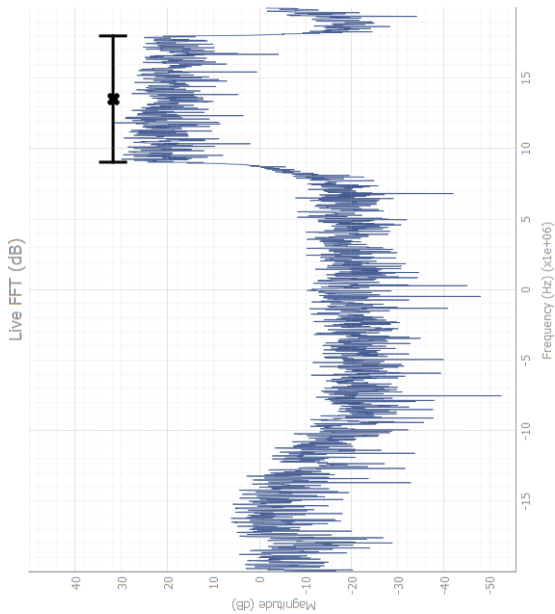
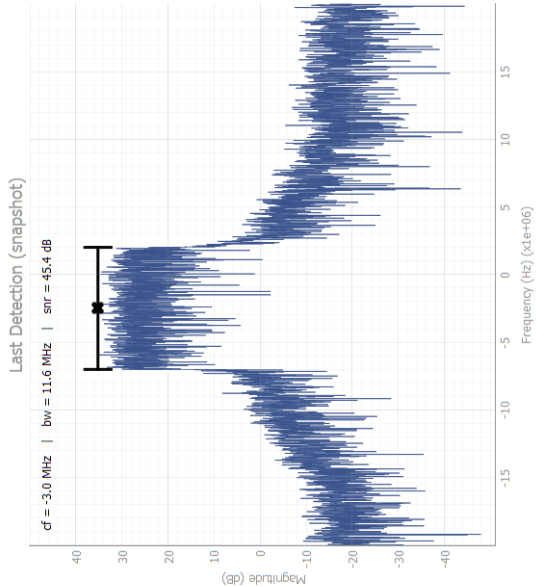


Figure 43: GUI, Theme: Night Ops



Figure 44: GUI, Theme: Resin

Status: Recorder: trigger ignored (still processing previous capture)



Parameters

Gain	Sample Rate	Center Freq (Hz)	Bandwidth (MHz)	SNR (dB)	Classification	Drone Count	Detected	Peak (dB)	Noise Floor (dB)	ResNet Top-2	SDR Tuned (MHz)
40.0	40000000	-2.988 MHz	11.562 MHz	45.37 dB	DJIMAVICAIR2: 85.1%	2	YES	35.21	-10.15	DJIMINI4PRO: 9.3%	2450.500

Throttle: 1.0d

STFT Display: Live STFT

Last FFT shows: Drone only

SDR Center Freq: 2.450 GHz

Custom Freq: 2450.500 MHz

Live FFT: Default

Show developer log

Log

System Log

[20:57:41] Recorder: trigger ignored (still processing previous capture)
[20:57:41] Recorder: trigger ignored (still processing previous capture)
[20:57:43] ResNet18: DJIMAVICAIR2 (85.1%)
[20:57:43] LATENCY AVG trigger—displayed:2439.9 ms (l=8)
[20:57:43] Split classifier: classified 2 drone spectrogram crops
[20:57:43] Recorder: CLASSIFIED -> DJIMAVICAIR2 (85.1%)
[20:57:43] STFT Time Total: 0.39696216583251953
[20:57:43] Recorder idle -> auto retune unblocked
[20:57:43] Recorder: buffers cleared after classification (dropped 0 queued chunks)
[20:57:43] Recorder busy -> auto retune temporarily blocked
[20:57:44] Recorder: trigger ignored (still processing previous capture)

Environment / Developer

Type	CF (MHz)	Rel CF (MHz)	BW (MHz)	SNR (dB)	Tre
DJIMINI4PRO (78%)	2447.786	-2.714	10.089	43.4	↑
DJIMAVICAIR2 (99%)	2463.942	13.442	9.169	42.8	→
CONTROLLER	-	-	-	43.8	↑
BLUETOOTH	-	-	-	42.8	↑

Emitter Log

Center Freq	Bandwidth	Confidence	Detection Rate
[20:57:41] DRONE	9.2	CERTAIN	6.50
[20:57:42] DRONE	9.5	CERTAIN	6.00
[20:57:42] DRONE	9.1	CERTAIN	5.50
[20:57:43] DJIMINI4PRO	9.8	conf= 78.0% split lId=15 abs=2448.2MHz	
[20:57:43] DJIMAVICAIR2	12.9	conf= 98.8% split lId=17 abs=2463.4MHz	
[20:57:43] DRONE	9.1	CERTAIN	5.00
[20:57:43] DRONE	9.1	CERTAIN	4.50
[20:57:43] DRONE	13.5	CERTAIN	5.00
[20:57:43] DRONE	-2.7	CERTAIN	5.00
[20:57:44] DRONE	13.4	CERTAIN	5.00
[20:57:44] CONTROLLER	-19.0	-	-
[20:57:44] DRONE	-3.0	CERTAIN	5.00

Figure 45: Detailed Current GUI Screenshot (Theme = Default Light)

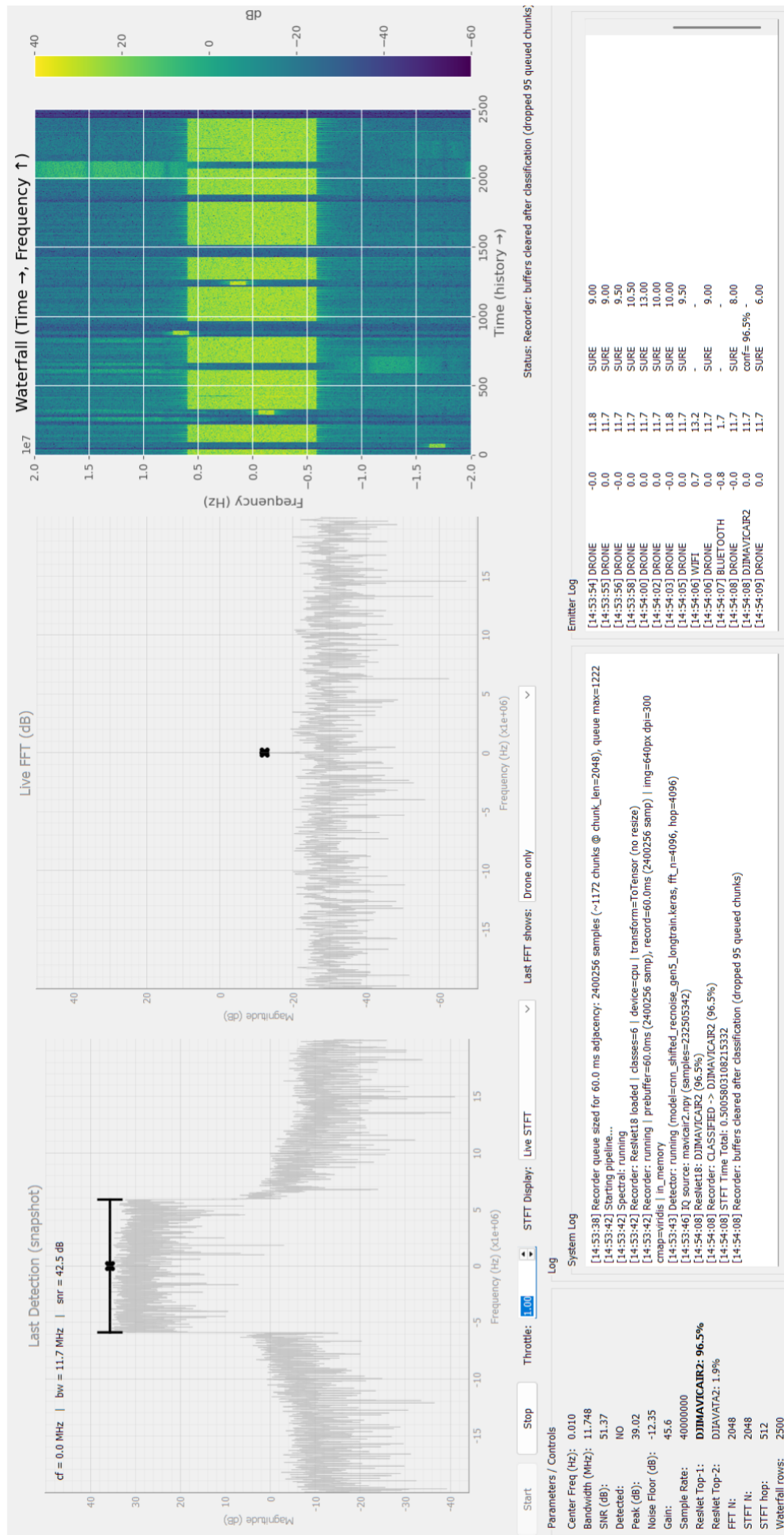


Figure 46: GUI at the Time of CM3

13 Detailed Timing Analysis of the GUI

This appendix features the timing analysis relating to the different parallel processes in the GUI.

Table 19: RecorderWorker Average Latency Breakdown

Stage	Average Latency (ms)
First Sample → Displayed	1818.8
First Sample → Last Sample	395.1
Last Sample → STFT Done	412.9
STFT Done → PNG Ready	622.0
PNG Ready → Model Input	3.6
Model Input → Model Done	331.1
Model Done → Classification Ready	0.9
Classification Ready → Displayed	53.2

Table 20: DetectorWorker Average Latency Breakdown

Stage	Average Latency (ms)
Chunk → Params	127.43
Chunk → Frame	14.33
Frame → FFT	3.80
FFT → Numeric Method Done	4.55
Numeric → Model Done	103.90
Model Done → Params	0.85

Table 21: MultipleDetectionWorker Average Latency Breakdown

Stage	Average Latency (ms)
Chunk → Emit	2849.00
Chunk → Input	12.20
Input → Burst Ready	2.21
Burst Ready → Processing Start	0.11
Processing Start → Processing Done	2834.47
Processing Done → Track Detection Done	0.02
Track Done → Emit	0.00

Table 22: Latency Breakdown of Average Other Detection Related Processes

Process	Average Latency (ms)
Processing Time for 1 FFT-Frame (signal absent)	10.1
Processing Time for 1 FFT-Frame (signal present)	16.6
Possible Detection $\rightarrow$ Certain Detection	30

14 Terminology Relating to the Project

- **Binary Decision Vector:** A vector consisting of 1's and 0's comparing each element's value with a given threshold. If each element is larger than the threshold, the entry corresponding to that element is 1, 0 otherwise.
- **Buffer:** A temporary storage that is capable of storing IQ data to be processed. Drops oldest frames if new data arrive while the buffer is full to include new data.
- **Burst:** Interleaved but continuous transmission or reception of data.
- **Chunk:** A part of data usually corresponding to a set length in samples or seconds.
- **Contaminated Frame:** A incorrectly labeled frame in the training dataset.
- **Emitter:** Signal transmitters and their representations in signals.
- **Emitter Isolation:** Isolation of each emitter to be processed individually.
- **Emitter Mask:** A vector consisting of only 1's and 0's that has entries of 1 where signals are present in the spectrum or the spectrogram.
- **Emitter Track:** The center frequency corresponding to where an emitter is consistently present.
- **FFT Frame:** The spectrum for a set number of samples, usually to be processed individually.
- **Frame:** A vector consisting of set number of samples.
- **IQ Data:** In-phase and Quadrature Data is real data that has been reduced to baseband to occupy less bandwidth with complex representation to carry the same information.
- **Isolated Spectrum:** Spectrum that has been artificially manipulated to isolate only a single emitter's data.
- **Main Thread:** The main process of the GUI that all other processes revolved around.
- **Model Inference Frame:** The frame that the neural network's decision has been made on. Both for STFT and FFT based models.
- **Noise Replacement:** Replacing certain parts of spectrums or spectrograms with artificially selected noise.

- Per-Bin Classification: Segmentation of the FFT signals into individual bands, with the emitter types specified.
- Per-Emitter Classification: Classifying each emitter in the band separately by isolation.
- Persistent Emitter State: Preserving each emitter's track for a set period of time since last detection to account for burst signals and duty cycles.
- Segment: A part of a signal vector that has been categorised on a similarity with nearby elements.
- Spectrogram Frame: A single spectrogram matrix.
- Spectrogram Image: A single spectrogram image.
- Threshold Crossing: Each element for a vector that is greater than a given threshold.
- Track Cooldown: The period of time where signal permanence is assumed.
- Track Merging: The merging of two signal tracks where the bandwidths of each overlap.
- Worker Thread: Parallel threads hosting each process independently.