A Navigation Signals Monitoring, Analysis and Recording Tool: Application to Real-Time Interference Detection and Classification

A Navigation Signals Monitoring, Analysis and Recording Tool: Application to Real-Time Interference Detection and Classification

Iman Ebrahimi Mehr, Alex Minetto, Fabio Dovis
Politecnico di Torino, Department of Electronics and Telecommunications, Turin, Italy

BIOGRAPHY

Iman Ebrahimi Mehr received the B.Sc. in electronic engineering from Azad University, Tehran, Iran, and his M.Sc. Degree in ICT for smart society from Politecnico di Torino, Turin, Italy, in 2012 and 2021, respectively. He is currently working on his Ph.D. in the Navigation Signal Analysis and Simulation Group, Department of Electronics and Telecommunications, Politecnico di Torino, Torino, Italy. His research interests include Global Navigation Satellite Systems (GNSS) and artificial intelligence applied to positioning, navigation and timing as well as to interference detection and mitigation.

Alex Minetto received the B.Sc. and M.Sc. degrees in Telecommunications Engineering from Politecnico di Torino, Turin, Italy and his Ph.D. degree in Electrical, Electronics and Communications Engineering, in 2020. He joined the Department of Electronics and Telecommunications of Politecnico di Torino in 2019 as research and teaching assistant and in 2022 as assistant professor. In 2015, he was an intern at European Organization for the Exploitation of Metereological Satellites (EUMETSAT), Darmstadt, Germany. His current research interests cover signal processing and advanced Bayesian estimation applied to Global Navigation Satellite System (GNSS) in space and critical infrastructure.

Fabio Dovis received the M.Sc. and Ph.D. degrees from the Politecnico di Torino, Turin, Italy, in 1996 and 2000, respectively. In 2004, he joined the Department of Electronics and Telecommunications, Politecnico di Torino as an Assistant Professor where he has been a full professor since 2021. He coordinates the Navigation Signal Analysis and Simulation Research Group. His research interests include design of GPS and Galileo receivers and advanced signal processing for interference and multipath detection and mitigation, and also ionospheric monitoring. Dr. Dovis is a Member of the IEEE Aerospace and Electronics Systems Society Navigation Systems Panel.

ABSTRACT

Given the extensive dependency on Global Navigation Satellite Systems (GNSS) for several crucial applications, the disruption caused by intentional or unintentional Radio Frequency Interference (RFI) may dramatically affect reliability and poses potential threats to various operations dependent on such systems. Recently, these threats have increased, and their detection and mitigation are of utmost importance in the field. To this aim, this paper presents an architecture for real-time detection and classification of RFI affecting multi-band GNSS signals based on a machine learning method. This study proposes an architecture combining an actual GNSS monitoring station for recording GNSS signals (Navigation Signals Monitoring, Analysis, and Recording Tool (N-SMART) system) with a deep neural network approach to detect and classify different classes of interferences. The proposed approach enables continuous monitoring, recording, and prompt alerting of RFI occurrences in multi-band GNSS signals, by leveraging the flexibility of a Software Defined Radio and docker frameworks. The design and deployment aspects of the proposed architecture are discussed, and the performance of the classification algorithm is evaluated. The results of the experimental test campaign on real interfered GNSS signals showed an overall accuracy of 85% and they highlighed the potential for effective, real-time classification of RFIs in GNSS.

I. INTRODUCTION

One of the crucial infrastructures in the modern world is Global Navigation Satellite Systems (GNSS), which are widely employed in a growing number of industries and have become essential to everyday life. The GNSS signals are vulnerable to Radio Frequency Interference (RFI) which can be a source of error for the signal processing stages of receivers, causing degradation in the estimation of Position, Velocity, and Time (PVT) or even complete losses of signal tracking, up to denials of service. Recording GNSS signals during instances of RFIs can prove valuable in studying their characteristics and extract the features that can then be utilized to develop mitigation solutions. Therefore, to ensure GNSS reliability to the users and mitigate the impact of RFI on GNSS signals, interference monitoring, detection, and classification are fundamental tasks. Several research studies have recently focused on monitoring interference signals, both from space (Murrian et al., 2019, 2021; Clements et al., 2023) and on the ground (Miguel et al., 2023; van der Merwe et al., 2023; Ward, 2007).

GNSS signals are not only subject to deliberate or unintentional interferences such as jamming and spoofing but also to ionosphere scintillation, multipath, and other signal anomalies (Spilker Jr et al., 1996). Storing high-fidelity radio-frequency signal samples enables a more detailed analysis of GNSS navigation signals, particularly during unique events or peculiar circumstances (Pica et al., 2023). The use of record and playback (RP) systems in the test and validation of GNSS receivers is growing due to their capability to replicate previously recorded live signals faithfully. Numerous studies in the literature have proposed using RP systems to evaluate GNSS receivers (Chu et al., 2010; Ilie et al., 2011, 2009). The utilization of RP systems is motivated by further several factors. Firstly, while high-end GNSS signal simulators or generators provided by various companies exist, they tend to be costly and burdensome. Secondly, in strongly constrained environments like city centers, the variability of urban scenarios may turn analytical models and simulators not representative of the complexity, and they show limited validity and scalability of the generated signals (Cristodaro et al., 2018; Richter et al., 2013). A cost-effective approach for deploying monitoring stations involves the utilization of Software Defined Radio (SDR) techniques. This approach enables the capture of signal samples that can be processed in real-time and subjected to post-processing for more extensive analysis. Additionally, in certain situations, it allows for replicating the scenario itself (Cristodaro et al., 2018; Linty et al., 2018; Brown et al., 2013; Mehr et al., 2023). Monitoring GNSS signals has relied on established techniques involving real-time signal processing using front-ends and dedicated monitoring receivers. Similar architectures can be found in the literature, where multi-frequency software receivers have been developed specifically for monitoring ionospheric scintillation (Curran et al., 2014; Peng and Morton, 2013; Linty et al., 2018; Pica et al., 2023).

Machine Learning (ML) methods have gained significant attention in GNSS applications in recent years, offering the potential to introduce new solutions and services. Limiting our interest to RFI detection and classification, numerous studies have explored the utilization of ML for this purpose. In certain studies, researchers have investigated features extraction from the time series of digital samples obtained from the receiver front-end output and applying ML methods like K-nearest neighbor (Qin and Dovis, 2022), Support Vector Machine (SVM) (Nicola et al., 2020), and Random Forest (RF) (Xu et al., 2020). Alternatively, other research studies have explored the utilization of Time-Frequency Representation (TFR) extracted from raw digital samples of signals. By leveraging TFR, the classification task can be transformed into an image classification task, enabling the application of deep learning models in this domain. These research studies (Ferre et al., 2019; Swinney and Woods, 2021; Liu et al., 2021; van der Merwe et al., 2023; Mehr and Dovis, 2023) demonstrate that Convolutional Neural Networks (CNNs) can be a valuable tool for RFI classification. Moreover, the research (Brieger et al., 2022) proposed an architecture that handles both time series and spectrogram data, utilizing CNN architecture and the TS-Transformer.

Despite existing solutions, there is a crucial requirement for a compact and flexible system delivering a reliable and autonomous collection of GNSS signal data, as well as automated detection and classification of interference.

This research proposes a novel architecture for real-time interference detection and classification of RFI, which can continuously monitor and record multi-band GNSS signals and provide timely warnings in case of RFI. The proposed architecture utilizes the Navigation Signals Monitoring, Analysis, and Recording Tool (N-SMART) system to grab and store the GNSS signals, while detection and classification are implemented based on a deep neural network technique. The basic principle of the suggested method is to implement a CNN classifier inside a Docker container and runs on top of the N-SMART system. The remainder of this paper is organized as follows: Section II introduces the N-SMART and the ML model. Section III discusses the proposed architecture and its deployment, detailing the use of Docker containers and the N-SMART system. Section IV presents the evaluation of the ML model's performance, highlighting the accuracy and inference time in real-world scenarios. Finally, Section V concludes the paper, summarizing our findings and offering insights into future lines of research and improvement.

II. BACKGROUND

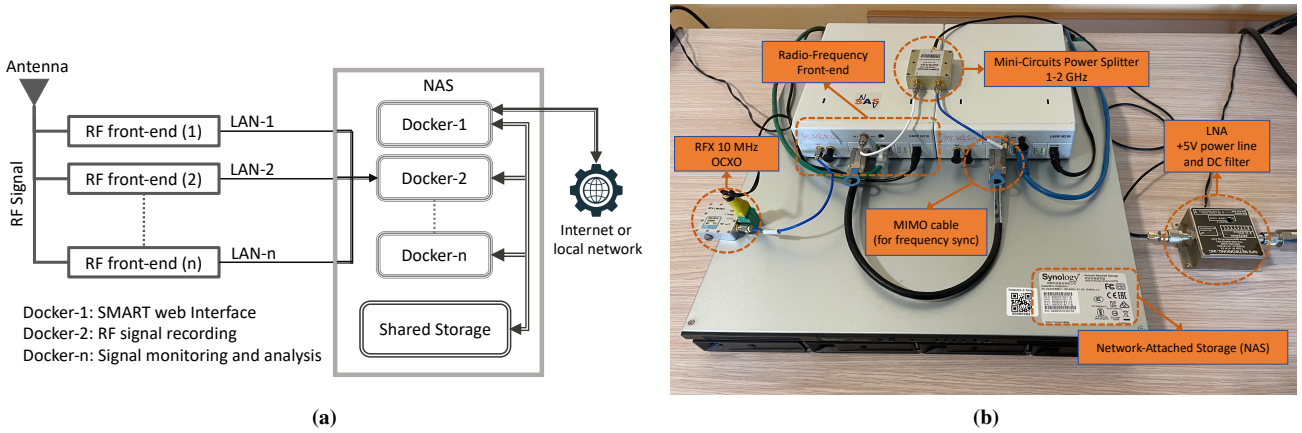
1. Navigation Signals Monitoring, Analysis and Recording Tool

N-SMART (Mehr et al., 2023) is an open architecture for GNSS signals monitoring, analysis, and a reliable, high-fidelity recording of the GNSS signals. It is designed to overcome the limitations of previous systems that struggled with data losses during the transfer between the front-end and the storage unit, particularly when high sampling frequency and high quantization depth are required. This system capitalizes on the microservices architecture, which organizes the entire application into a series of small, autonomous services. *Containers* are a technology that significantly simplifies the development and deployment of microservices. This architecture uses Docker containers to package the software components of the system, including the application for signal analysis and recording and its required dependencies. A *Docker container* is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime environment and variables, libraries, and config files. Figure 1a depicts a high-level of the N-SMART system block diagram, outlining both the hardware and software components. As depicted in Figure 1a, a specific Docker container is dedicated to the task of GNSS signal recording. This container communicates with the radio frequency front-end devices to capture the signal through an ethernet connection. This Docker container hosts the Universal Software Radio Peripheral (USRP) Hardware Driver (UHD), integral to the front-end (USRP N-210) communication. Additionally, it contains the pipeline for grabbing the signal, which has been designed using GNU Radio. Moreover, another Docker container is responsible for hosting the web User Interface

(UI) application, which can be accessed over the internet. This UI enables users to monitor the system status, reconfigure the system settings, analyze recorded signals (spectrum and spectrogram), and control GNSS signals acquisition, providing a comprehensive and accessible tool for managing and interacting with the system.

The utilization of Docker containers allows the application to be installed on any hardware system without requiring prerequisite installations. N-SMART is installed on a Network Attached Storage (NAS) unit, which is a standalone data storage system connected to a network and acts as a central data repository. N-SMART is a fully open-source and configurable system, and the compact implementation of the data processing in the NAS makes this architecture more suited for in-field implementation and monitoring operations to be performed in remote regions of the Earth lacking or with minimal power supply and connectivity resources. N-SMART can capture multi-frequency of GNSS band and provides Intermediate-Frequency (IF), In-phase/Quadrature (IQ) signal samples. Furthermore, the recorded signals by N-SMART contains the GNSS software-defined receiver metadata standard (Gunawardena et al., 2021), which defines parameters and schema to express the contents of the recorded dataset. Figure 1b shows the hardware installation and components of the N-SMART system and its configuration in operation at the NavSAS laboratories.

The motivation behind utilizing the N-SMART system lies in its dockerization architecture, which is well-suited to recent advancements in signal analysis and has the capability to host a separate Docker container, such as ML deployment, that are specialized for on-site monitoring algorithms. Providing direct access to the real-time stream of GNSS signals within the guest container, eliminates the necessity of transferring large volumes of SDR-captured data to remote processing centers.



(a) **(b)**

Figure 1: High-level hardware and software block diagram (a), and Prototype of an operational GNSS Monitoring and Grabbing Station at the NavSAS laboratories (b).

2. Applying Machine Learning to the Real World.

ML aims to solve complex tasks and make informed decisions by utilizing algorithms that enable computers to learn patterns and insights from data. ML models can provide valuable predictions, recommendations, or classifications by analyzing and processing vast amounts of information, improving problem-solving and decision-making capabilities. In the context of solving a task or making decisions, the ML pipeline can be broadly divided into two main steps: *development* and *deployment* (Raj, 2021). The process of deploying machine learning models into production is illustrated in Figure 2.

- **ML Development:** The development process of ML begins with the collection and preparation of data. This step involves gathering the necessary data, cleaning it, addressing missing values, and transforming it into a suitable format. Once the data is ready, the subsequent step is to select a suitable ML algorithm or model architecture that aligns with the specific problem being addressed. The chosen model is then trained using the prepared data to identify patterns, establish relationships, and ultimately make accurate predictions or classifications based on the learned patterns. Following the model training phase, a crucial step in the ML development process is evaluating the model to measure its performance and uncover any limitations. Metrics like accuracy, precision, recall, and inference time are utilized to measure its performance against the desired objectives. If necessary, optimization techniques like hyperparameter tuning are applied to enhance its performance (Flach, 2012).
- **ML Deployment:** This step entails deploying the models into a production environment, making them accessible and available for end users, and allowing them to be used for real-world applications and decision-making. ML deployment is essential because it bridges the gap between model development and practical use, enabling organizations to leverage the

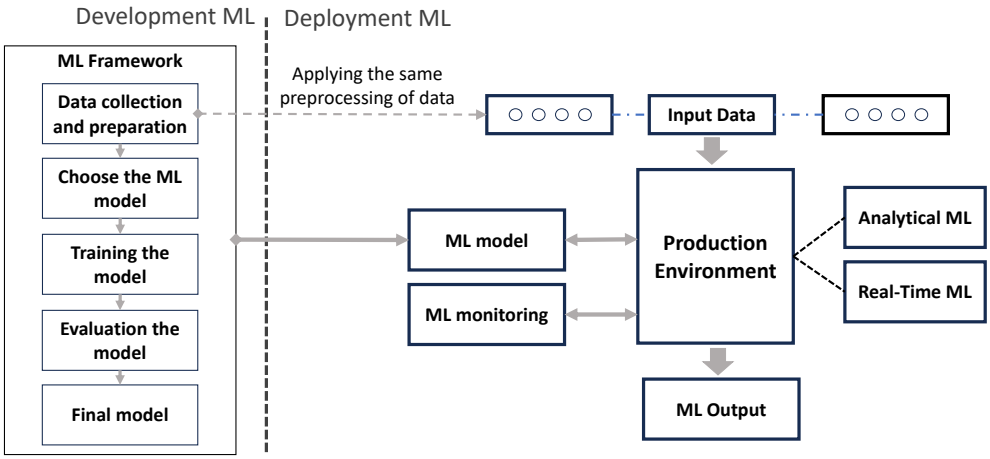


Figure 2: From ML model to Production

benefits of ML in their day-to-day operations while minimizing costs and maximizing efficiency. During the deployment phase, it is necessary to establish the required infrastructure, which may involve configuring servers or utilizing cloud platforms to host and serve the model. Continuing the development process, it is essential to monitor the performance of the deployed model, ensuring its ongoing accuracy and reliability. Additionally, based on user feedback or changes in the problem domain, updates or improvements to the model might be necessary ((Bhatt et al., 2020); (Baier et al., 2019)). Deployment of ML can indeed be categorized into two main types: analytical ML and real-time ML. Analytical ML typically operates in an "offline" environment, where the model analyzes historical data and generates insights or predictions such as data mining or generating reports. Real-time ML integrates models into applications to facilitate autonomous and continuous decision-making, directly influencing real-time business operations. This deployment scenario enables applications to dynamically and promptly respond to incoming data, leveraging the output of ML model to make immediate predictions or take actions (De Bie et al., 2 23).

The selection of appropriate machine learning models, frameworks, and tools is vital, especially in resource-limited areas like the GNSS domain, where hardware and software resources are scarce. The choice of frameworks and tools for ML deployment should be based on the target environment's specific requirements, constraints, and available resources to ensure optimal performance, efficiency, and compatibility. For instance, if the ML model is intended to run within a GNSS receiver or a mobile device with limited resources, it may require lightweight frameworks and optimized algorithms to ensure efficient execution. On the other hand, if the model is intended to run in a more powerful environment, such as a dedicated monitoring station, there may be more flexibility regarding the frameworks and tools used, allowing for more resource-intensive options. While ML techniques can be computationally intensive and time-consuming, GNSS applications often demand real-time or near-real-time processing, particularly for positioning and navigation tasks. Hence, a framework or tool for ML deployment can be deemed efficient based on its capacity to optimally utilize resources like memory, CPU, and time, enabling seamless integration with GNSS applications. In addition, the popularity and support of the framework need to be considered, where popularity reflects the widespread usage, well-regarded, and active support within the developer community, whether the framework is open-source or closed-source.

3. Deep Neural Network Approach for detection and classification

RFIs, even intentional from jammers or unintentional, can cause disruptions to GNSS receivers within a specific operational area, potentially rendering them non-functional. In our previous research (Mehr and Dovis, 2023), we introduced an ML technique, referred to as the feature-aided CNN classifier, which leverages deep neural networks to automatically and accurately detect and classify instances of interferences in GNSS band. The conceptual block diagram for this method is depicted in Figure 3. The input to this system is the stream of raw samples of the received GNSS signal, which may contain potential interference. During the preprocessing stage, pertinent information for the model classifier is extracted by analyzing the input signal across various domains. Initially, the input signal (a window of $100\,\mu s$) is examined within the time-frequency domain, creating a TFR that is preserved as an image. Concurrently, additional relevant features, as listed in Table 1 are extracted from both the time and frequency domains of the same time series of input signals (a window of $200\,\mu s$) as numerical features. Then the image generated by the TFR is processed through a CNN to extract the most significant and informative features. Subsequently, these extracted features are merged with numerical features. This fusion of diverse features forms a combined feature vector, which is then inputted into the fully connected layers of the CNN for the purpose of classification. The final layer of the

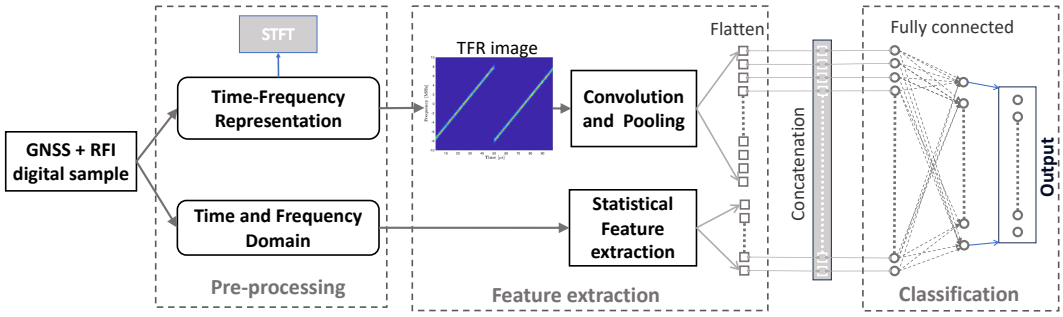


Figure 3: Feature-Aided CNN classifier

model is a softmax layer, which provides the probabilities associated with each interference type. TFR can be derived using several transformations, including the Short-Time Fourier Transform (STFT) and the Wigner-Ville Transform. Our preliminary investigation (Mehr and Dovis, 2023) revealed that while these TFR methods offer nearly equivalent accuracy, the STFT is more computationally efficient. Furthermore, it is also shown that ResNet has better accuracy with respect to AlexNet architecture. As a result, we have chosen to utilize the STFT for generating the TFR and ResNet for the CNN architecture in this context.

Table 1: List of features in time and frequency domains

Domain	Features	Description
Time Domain	Mean value Median value Standard deviation Mean absolute deviation Root Mean Square 25th percentile 75th percentile Inter Percentile Range	Common statistical values of raw sample to measure the dispersion of the samples
	Skewness	Measure the asymmetry of the probability distribution.
	Kurtosis	Measure the outlier-prone of the probability distribution.
	Entropy	Measure the uncertainty and randomness of the samples.
Frequency Domain	Frequency of max power	Frequency of the maximum power located.
	Maximum Power	Maximum power obtained in frequency domain.
	Mean Power	Mean power obtained in frequency domain.

III. METHODOLOGY

The research introduces an innovative architecture for the real-time detection and classification of RFI in multi-band GNSS signals. The primary objective of this architecture is to continuously monitor and record the GNSS signals while promptly alerting potential RFI occurrences. The proposed approach leverages the N-SMART system for capturing and storing the GNSS signals, while the detection and classification tasks are performed using the feature-aided CNN classifier that operates within a Docker container. This container is integrated with the N-SMART system, allowing the CNN classifier to get the real-time GNSS signals as inputs. By combining these two functionalities, the proposed architecture offers a comprehensive solution for continuous recording and timely warning of RFI in multi-band GNSS signals.

1. Real Data Integration

a) Feature-aided CNN classifier with simulated data

In our previous work (Mehr and Dovis, 2023, 2022), the result of the feature-aided CNN classifier demonstrated its potential as a robust tool for interference classification. The obtained accuracy was 98% when evaluated on a simulated dataset comprising 17 classes in which one class represented the GNSS signal without interference. While the remaining 16 classes represented GNSS signals in the presence of various types of interference, including the most common chirp signals used by jammers and documented in (Pattinson et al., 2017), Frequency Hopping (FH) jammers, Continuous Wave Interference (CWI), narrowband jammers, Frequency Modulation (FM) and Amplitude Modulation (AM) jammers. In the simulated scenario, the GNSS signals belong to the L1 band and originate from the GPS constellation. These signals exhibit a carrier-to-noise ratio (C/N_0) ranging

from 25 to 50 dBHz, where the background noise level is set to -202 dBW/Hz. An 8th-order Butterworth filter with a bandwidth of 20.46 MHz is employed to process the GPS signal. The resulting signals are then sampled at a rate of 40 MHz in the I/Q baseband without considering any quantization bit. To simulate the presence of interferences, various types of interference signals are combined with the GNSS signals by adding them together. The interference power levels range from -142 dBW to -107 dBW, increasing in increments of 1 dBW. Consequently, the interference-to-noise ratio spans from -13 dB to 22 dB.

b) Learning the model with dataset from real scenario

To generate a dataset that reflects a real scenario, the recorded signals are obtained from N-SMART in the presence of various interferences. The L1 band of the GNSS signal is captured using a USRP N-210 device, which operates at a sampling frequency of 25 MHz and utilizes 16-bit quantization (I/Q). For transmitting the interference signals, another USRP device is employed and reads the corresponding .bin files containing different interference signals and transmits them within the GNSS L1 band. The same 16 classes of interferences and range power as in the case of the simulated scenario are used. Figure 4 demonstrates the configuration of N-SMART integrated into the real interference scenraio.

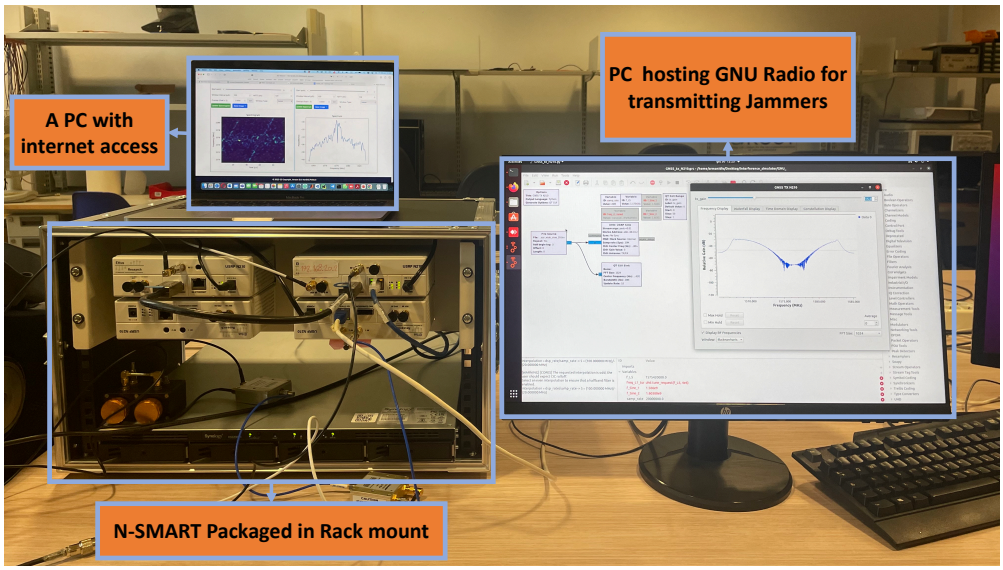


Figure 4: Configuration of N-SMART integrated into Real interference Scenario

Due to the differences in sampling frequency and bit quantization between the simulated and the real scenarios, retraining the model becomes necessary. Since training the model from scratch can be time-consuming and resource-intensive, a Transfer Learning (TL) approach is employed to facilitate retraining. TL is a ML technique where knowledge gained from training a model on one task is applied to a different but related task (Pan and Yang, 2010). This technique leverages the pre-existing knowledge and parameters of the model learned from the simulated data, reducing the time and computational resources required for training.

The development framework used for building CNN models is the TensorFlow library in Python. TensorFlow (Abadi et al., 2015) is an open-source library for ML and artificial intelligence, and it allows for building packages that can be deployed on various CPU architectures. The model's training process is performed in the Google Colaboratory (Colab) environment (Bisong, 2019). The Adam optimization algorithm is employed to train the model, and the initial learning rate used in the training process is set to 0.001. The epoch number of the training process is 25, and early stopping for termination of training in case of no improvement in validation loss is utilized. In addition, the adaptive learning rate is implemented to dynamically adjust the learning rate (Mehr and Dovis, 2023). Figure 5 illustrates the training performance on the validation dataset, demonstrating the impact of applying TL versus not using TL. It is observed that when TL is applied, the model achieves its lowest loss after only five epochs. In contrast, the model takes ten epochs without TL to converge and reach the lowest loss. Once the training process concludes, the optimal model is saved, including the complete model architecture, weights, and parameters, in order to serve as the model for real-time purposes.

2. Real-Time Deployment of Interference Detection and Classification

Real-time implementation of ML models has gained significant importance in various domains, enabling timely decision-making, automation, and intelligent processing of data streams. This section focuses on implementing the feature-aided

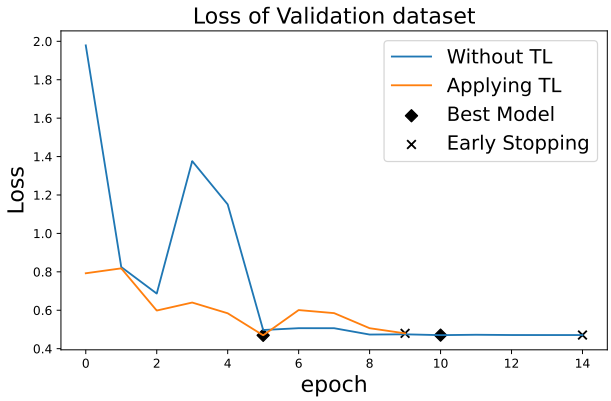


Figure 5: Training of Feature-Aided CNN classifier

CNN classifier for real-time detection and classification of RFIs. The real-time deployment involves continuously processing incoming data streams, enabling immediate analysis and decision-making. The implementation framework encompasses several key components as follows:

- Data acquisition and preprocessing: The process of acquiring data involves the continuous recording of GNSS signals using the N-SMART system, as explained in Section II.1. The system captures the GNSS signals and produces a continuous stream of IF, high-fidelity IQ signal samples. The system reads and transfers the most recent 200 microseconds of recorded signal to the preprocessing stage every second. The decision to perform analysis every second depends on the resource constraints of NAS device in the N-SMART system. In the data preprocessing step, the TFR (spectrogram) of the received signal and the corresponding statistical features are derived, serving as the ML model's input. As a metric of evaluation, the time needed to complete the preprocessing task, starting from reading data and preparing input of the model, is called preprocessing time.
- Data processing: The processing phase of model deployment is carried out within a separate and isolated Docker container which is created using the official TensorFlow Docker image and includes an added API. Through this API, the Docker container receives input data from the preprocessing step and returns the associated probabilities of the presence of each interference type. A script is written to perform the processing step, which involves loading the best-trained model from the training part and passing the input data to the model for inference tasks. It is essential to highlight that the CPU of the NAS device utilized in this experiment does not support Advanced Vector Extensions (AVX). Consequently, the precompiled version of the TensorFlow package, which does not support CPUs without AVX, cannot be used. To overcome this limitation, the TensorFlow package is built and compiled from the source code, ensuring compatibility with the NAS device's CPU configuration. Similar to preprocessing time, the time that model takes to apply a trained neural network to new data is called the inference time.
- Interference Monitoring: To facilitate monitoring, a dedicated webpage is created as part of UI, providing a visual representation of the ML model's performance and the corresponding classification results. Additionally, an automatic alert system is developed to ensure timely awareness of critical situations.

IV. EXPERIMENTAL FINDINGS AND ANALYSIS

This section presents the experimental results of the feature-aided CNN classifier for the detection and classification of interference. The model is evaluated in a real interference scenario, as explained in section III.1 b), and the obtained results are summarized in Table 2. The preprocessing, inference, and task completion are measured when the system is performed signals recording and the real-time classification task while the accuracy and F1-scores are related to the test dataset (offline mode). To ensure a fair comparison between the simulated and real scenarios, we created a simulation scenario in which we aligned the simulation parameters with those of the real scenario. Consequently, we employed a 25 MHz sampling rate and 16-bit quantization for the simulated data. The associated results are shown in Table 2.

The results presented in Table 2 demonstrate the effectiveness of the feature-aided CNN classifier in detecting and classifying with an overall accuracy of 85.13% across all interference types. It is noteworthy to compare these results with the accuracy achieved when the model is tested on simulated data. In the simulated scenario experiments, the classifier achieved an accuracy of 95.75%. However, when evaluated on a real dataset, the accuracy decreased to approximately 88%. The decrement in accuracy, i.e., 7%, when transitioning from simulated to real scenarios can be attributed to several factors. Firstly, while it is accurate that

Table 2: Result of Detection and Classification of Interference

Mode	Accuracy [%] (Test dataset)	F1-score [%] (Test dataset)	Pre-processing time [ms]	Inference time [ms]	Task completion time [ms]
Simulated scenario	95.75	95.43	87	77	164
Real scenario	87.76	87.23	412	275	687

the sampling frequency and bit quantization remain consistent across both scenarios, it is important to note that the front-end architectures differ, and each component within them can influence the data. Consequently, one can infer that, in a real-world implementation and deployment, the machine learning algorithm should be trained according to the specific front-end hardware, utilizing signal instances emerging from that particular front-end.. Secondly, real-world interference scenarios often contain variations and noise that might not be accurately represented in the simulated data. The presence of unmodeled or unknown factors in real scenarios can pose challenges for the classifier, where the model might not have encountered certain types of variations that exist in the real dataset, leading to a decrease in accuracy. Despite the decrease in accuracy, the classifier still maintains a reasonable performance level and indicates the potential of the feature-aided CNN classifier in practical applications for real-time interference detection and classification.

The preprocessing and inference time metrics are strongly influenced by the available resources on the target platforms. In the given experiment, the NAS system is equipped equipped by an INTEL Atom C3538-2.1 GHz CPU and 4 GB of RAM, while the simulation is executed on a personal computer equippedby an Intel Core i5- 3GHz and 4 GB of RAM. It is worth noting that utilizing a more powerful CPU can potentially reduce the time required to raise an alarm, as it can handle the preprocessing and inference tasks more efficiently.

Table 3 presents a Class-Wise analysis of the model's performance, offering insights into precision, recall, and F1-score. Precision measures correct positive predictions relative to all positive predictions, indicating the model's accuracy in positive classifications. Recall is the measure of correct positive predictions relative to all actual positives, reflecting the model's ability to capture all relevant instances, and the F1-score is a balanced metric that combines precision and recall and interference types with high F1-scores demonstrate good overall classification performance.

Table 3: Model performance metrics for various interference types

Interference type	Precision [%]	Recall [%]	F1-score [%]
GNSS (No interference)	81	83	82
CWI	85	77	81
DME (pulsed) interference	83	81	82
FH jammer	82	73	77
FMI	87	83	85
Hooked sawtooth	94	93	93
Multitone narrow	90	92	91
NarrowBnad Jammer	80	74	77
Linear narrow	92	88	90
Linear_wide_fast	91	90	90
Linear_wide_medium	94	95	94
Linear_wide_rapid	93	89	91
Linear_wide_slow	95	94	94
Sawtooth	93	89	91
Tick	96	94	95
Triangular wave	88	87	87
Triangular	85	82	83

V. CONCLUSION

In conclusion, this research effectively addressed the deployment of ML models for RFI detection and classification in GNSS. The proposed architecture demonstrates an overall accuracy of 88% for classification in a real scenario of interference signals and provides reliable, real-time monitoring and prompt alerts. In future work, there is potential to optimize the model or explore new architectures of CNN to enhance accuracy and reduce task completion time. Additionally, efforts can be made to expand the application of the model across a wider range of interferences.

ACKNOWLEDGEMENTS

The authors acknowledge the support from INGV and LINKS researchers and their contribution to its conceptualization. The Ph.D. work of I. Ebrahimi Mehr is supported by the grant DOT1332092 CUP E11B21006430005 funded within the Italian Programma Operativo Nazionale (PON) Ricerca e Innovazione 2014-2020, Asse IV "Istruzione e ricerca per il recupero" con riferimento all'Azione IV.4 "Dottorati e contratti di ricerca su tematiche dell'innovazione" e all'Azione IV.5 "Dottorati su tematiche green" DM 1061/2021. A. Minetto acknowledges funding from the research contract no. 32-G-13427-5 DM 1062/2021 funded within the PON Ricerca e Innovazione of the Italian Ministry of University and Research.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Baier, L., Jöhren, F., and Seebacher, S. (2019). Challenges in the deployment and operation of machine learning in practice. In *European Conference on Information Systems*.

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F., and Eckersley, P. (2020). Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, page 648–657, New York, NY, USA. Association for Computing Machinery.

Bisong, E. (2019). *Google Colaboratory*, pages 59–64. Apress.

Brieger, T., Raichur, N., Jdidi, D., Ott, F., Feigl, T., van der Merwe, J., Rügamer, A., and Felber, W. (2022). Multimodal learning for reliable interference classification in gnss signals. *Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, Colorado,*, pages 3210–3234.

Brown, A., Redd, J., and Dix, M. (2013). Open source software defined radio platform for gnss recording, simulation. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)*, pages 1508–1516, Nashville, TN.

Chu, T., Vinande, E., Akos, D., and Weinstein, B. (2010). Gnss receiver evaluation : record-and-playback test methods. *GPS World*, 21(1):28–34.

Clements, Z., Humphreys, T. E., and Ellis, P. (2023). Dual-satellite geolocation of terrestrial gnss jammers from low earth orbit. In *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 458–469.

Cristodaro, C., Ruotsalainen, L., and Dovis, F. (2018). Benefits and limitations of the record and replay approach for GNSS receiver performance assessment in harsh scenarios. *Sensors (Switzerland)*, 18(7).

Curran, J. T., Bavaro, M., Morrison, A., and Fortuny, J. (2014). Developing a multi-frequency GNSS-based scintillation monitoring receiver. In *27th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2014*, volume 2, pages 1142–1152.

De Bie, T., De Raedt, L., Hernández-Orallo, J., Hoos, H. H., Smyth, P., and Williams, C. K. (2022-02-23). Automating data science: Prospects and challenges.

Ferre, R., Fuente, A., and Lohan, E. S. (2019). Jammer classification in gnss bands via machine learning algorithms. *Sensors*, 19:4841.

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press.

Gunawardena, S., Pany, T., and Curran, J. (2021). Ion gnss software-defined radio metadata standard. *NAVIGATION: Journal of the Institute of Navigation*, 68(1):11–20.

Ilie, I., Fortin, D., and Fortin, M.-A. (2009). Multi-channel record and playback system for gnss rf/if receivers' design validation and fine-tuning. In *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*, pages 2265–2275, Savannah, GA.

Ilie, I., Hini, R., Cardinal, J.-S., Blood, P., and Fortin, D. (2011). Record and playback system for gnss: What you need to know for successful testing. In *Proceedings of the 24th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2011)*, pages 2009–2021, Portland, OR.

Linty, N., Dovis, F., and Alfonsi, L. (2018). Software-defined radio technology for GNSS scintillation analysis: bring Antarctica to the lab. *GPS Solutions*, 22(4).

Liu, Z., Lo, S., and Walter, T. (2021). Gnss interference detection using machine learning algorithms on ads-b data. In *Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021)*, pages 4305–4315.

Mehr, I. E. and Dovis, F. (2022). Detection and classification of gnss jammers using convolutional neural networks. In *2022 International Conference on Localization and GNSS (ICL-GNSS)*, pages 01–06.

Mehr, I. E. and Dovis, F. (2023). A deep neural network approach for detection and classification of gnss interference and jammer. doi: 10.36227/techrxiv.22212121.

Mehr, I. E., Minetto, A., Dovis, F., Pica, E., Cesaroni, C., and Romano, V. (2023). An open architecture for signal monitoring and recording based on sdr and docker containers: A gnss use case. In *IEEE EUROCON 2022 - 20th International Conference on Smart Technologies,*. doi: 10.36227/techrxiv.22212142.

Miguel, N. R. S., Chen, Y.-H., Lo, S., Walter, T., and Akos, D. (2023). Calibration of rfi detection levels in a low-cost gnss monitor. In *2023 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 520–535.

Murrian, M., Narula, L., and Humphreys, T. (2019). Characterizing terrestrial gnss interference from low earth orbit. pages 3239–3253.

Murrian, M. J., Narula, L., Iannucci, P. A., Budzien, S., O'Hanlon, B. W., Psiaki, M. L., and Humphreys, T. E. (2021). First results from three years of gnss interference monitoring from low earth orbit. *NAVIGATION, Journal of the Institute of Navigation*, 68(4):673–685.

Nicola, M., Falco, G., Morales Ferre, R., Lohan, E.-S., de la Fuente, A., and Falletti, E. (2020). Collaborative solutions for interference management in gnss-based aircraft navigation. *Sensors*, 20(15).

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pattinson, M., Lee, S., Bhuiyan, Z., Thombre, S., Manikundalam, V., and Hill, S. (2017). D4.2: Draft standards for receiver testing against threats. Technical report, STRIKE3 (European GNSS Agency).

Peng, S. and Morton, Y. (2013). A USRP2-based reconfigurable multi-constellation multi-frequency GNSS software receiver front end. *GPS Solutions*, 17.

Pica, E., Minetto, A., Cesaroni, C., and Dovis, F. (2023). Analysis and characterization of an unclassified rfi affecting ionospheric amplitude scintillation index over the mediterranean area. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 1–20.

Qin, W. and Dovis, F. (2022). Situational awareness of chirp jamming threats to gnss based on supervised machine learning. *IEEE Transactions on Aerospace and Electronic Systems*, 58(3):1707–1720.

Raj, E. (2021). *Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale*. Packt Publishing.

Richter, R., Wolf, B., and Michler, O. (2013). Evaluation of gnss rf signal simulators and receivers based on recorded multi gnss signals in scenarios of traffic telematics. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013)*, pages 1881–1889, Nashville, TN.

Spilker Jr, J. J., Axelrad, P., Parkinson, B. W., and Enge, P. (1996). *Global positioning system: theory and applications, volume I*. American Institute of Aeronautics and Astronautics.

Swinney, C. J. and Woods, J. C. (2021). Gnss jamming classification via cnn, transfer learning & the novel concatenation of signal representations. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–9.

van der Merwe, J. R., Contreras Franco, D., Hansen, J., Brieger, T., Feigl, T., Ott, F., Jdidi, D., Rügamer, A., and Felber, W. (2023). Low-cost cots gnss interference monitoring, detection, and classification system. *Sensors*, 23(7).

Ward, P. (2007). What's going on?: Rfi situational awareness in gnss receivers. *InsideGNSS*, 2(6):35–42.

Xu, J., Ying, S., and Li, H. (2020). Gps interference signal recognition based on machine learning. *Mobile Networks and Applications*, 25.