uc3m | Universidad Carlos III de Madrid

Master Degree in Telecommunications Engineering
2024-2025

Master Thesis

A Data-Driven Framework for Wireless Communications

Mario Golbano Corzo

Supervisor
Alejandro Lancho Serrano

Madrid, 2025 July

SUMMARY

This thesis presents the development of a comprehensive dataset for wireless communication signal processing, aimed at supporting machine learning applications in signal classification, interference detection, and general signal processing. A key distinction of this dataset is that the signals are not generated from random bitstreams, as is common in most related works, but rather from the actual encoding of real video files into binary form. This design choice makes the dataset highly relevant for practical applications and allows for advanced evaluation scenarios in which model performance can be analyzed not only at the signal level but also at the application level (e.g., video reconstruction). Moreover, it opens the door to future research in areas such as semantic communications, where the goal is to preserve meaning rather than strictly bit-level accuracy.

To generate this dataset, a suite of custom MATLAB applications was developed. These include tools for video encoding into bitstreams, wireless signal generation with diverse modulation schemes, and additional modules for signal demodulation and visualization. These tools allow end-to-end evaluation of any model trained on the dataset. In parallel, Python tools were created for visualizing the generated signals, providing insight into their characteristics and aiding in the evaluation of their suitability for machine learning tasks.

While the primary focus of this work is the creation of the dataset, a machine learning model, specifically the UNet architecture, is also applied to the dataset to assess its viability for signal processing tasks. The results of this preliminary evaluation demonstrate the potential of the dataset for future research in data-driven wireless communication systems, offering a valuable resource for advancing the field of machine learning-based signal processing. This work lays the foundation for future studies exploring more advanced machine learning models for interference mitigation, signal separation, and wireless communication optimization.

Keywords: *Wireless Communications, RF (Radio Frequency), ISM (Industrial, Scientific and Medical) Band, Dataset, Modulation, Interference, Interference Mitigation, Machine Learning.*

DEDICATION

To Juan, my family and friends and my thesis supervisor, Alejandro.

Contents

List of Figures

List of Tables

1. Introduction

Wireless communications have been a cornerstone of modern society since the early 20th century, evolving from basic radio signals to the complex mobile networks and IoT systems we rely on today. As the demand for higher capacity, lower latency, and more robust networks continues to grow, the integration of artificial intelligence (AI) into wireless technologies presents groundbreaking opportunities. AI promises to optimize network management, enhance signal processing, and address persistent challenges such as interference and spectrum management, paving the way for smarter, more adaptive communication systems and the next generation of self-optimizing networks.

To contextualize the contributions of this work, the first subsection of the introduction is devoted to a review of the current state of the art in machine learning applications for wireless signal processing. In particular, it focuses on existing datasets that support RF signal classification as well as other tasks related to this subject, which directly motivate the creation of the dataset proposed in this thesis.

1.1. State of the Art

1.1.1. General Datasets for RF Signal Classification and Processing

This subsection presents widely known datasets designed for RF signal classification and processing tasks. These datasets include a variety of modulation schemes and signal types and are specifically created for different wireless signal processing applications.

The *RF Challenge: The Data-Driven Radio Frequency Signal Separation Challenge*

The *RF Challenge: The Data-Driven Radio Frequency Signal Separation Challenge* is an initiative that promotes the use of deep learning methods for source separation and interference rejection in radio-frequency (RF) signals [1]. The challenge focuses on signal interference scenarios where co-channel interference overlaps in both time and frequency with the signal of interest (SOI). It aims to advance the development of robust algorithms that can handle the intricacies of RF signals without relying on predefined models of the interference. This dataset consists of various real-world RF signal mixtures and provides a benchmark for testing data-driven interference mitigation approaches.

Dataset Details:

- **QPSK and OFDM-QPSK Signals**: The dataset includes single-carrier QPSK and multi-carrier OFDM-QPSK signals as SOIs.
- **Interference Signals**: Four types of interference signals are included—EMISignal1, CommSignal2, CommSignal3, and CommSignal5G1. These interferences are recorded

from actual RF devices and are characterized by their non-Gaussian nature and varying bandwidths. The generation process for some of these interference signals is unknown, while for others, such as CommSignal5G1, it is known.

The *RF Challenge* dataset enables testing of interference rejection techniques, particularly for environments where interference is complex and unpredictable, and where the signal generation process may or may not be available.

The *RadioML 2018.01A* Dataset: A Benchmark for RF Signal Classification

One of the most prominent datasets in the field of machine learning for wireless communications is *RadioML 2018.01A*, introduced by O'Shea et al. in their influential work on over-the-air (OTA) signal classification [2]. This dataset serves as a cornerstone for benchmarking radio signal classification models, especially those leveraging deep learning.

RadioML 2018.01A consists of 24 distinct modulation classes, including a mix of analog and digital schemes, and both low- and high-order modulations such as AM, FM, GMSK, OOK, BPSK, QPSK, and up to 256-QAM and 128-APSK. The dataset is split into two configurations: a "Normal" set (11 basic modulations) and a more challenging "Difficult" set (all 24 classes), designed to emulate realistic classification complexity. Each sample consists of 1024 I/Q time-domain samples, simulating a short-time observation typical in many real-world RF tasks such as burst signal classification or rapid scanning scenarios.

The signals are synthetically generated using GNU Radio-based simulation chains, incorporating realistic channel impairments such as additive white Gaussian noise (AWGN), carrier frequency and clock offset, symbol rate offset, and multipath Rayleigh fading. It is worth noting, that these signals were generated based on purely random bitstreams, not reflecting the structure nature of many real-world transmissions.

This combination of synthetic diversity and impairment realism positions *RadioML 2018.01A* as a robust dataset for evaluating RF classifiers under challenging signal conditions.

The confusion matrices presented in the paper show frequent misclassifications among high-order modulations (e.g., 64-QAM vs. 256-QAM) and among analog AM variants. These results highlight the difficulty of discriminating between spectrally dense modulations, especially in low SNR environments.

The *WAIR-D* Dataset: Wireless AI Research Data for Realistic Channel Modeling

The Wireless AI Research Dataset (*WAIR-D*) [3] represents a significant step forward in the development of realistic datasets for wireless communication systems enhanced by machine learning. Proposed by researchers from Huawei Technologies and Zhejiang

University, *WAIR-D* was specifically designed to address the limited diversity and realism found in existing wireless datasets, which are typically based on statistical models or constrained ray-tracing simulations.

WAIR-D distinguishes itself by combining large-scale data generation with high environmental fidelity. It is constructed using the PyLayers 3D ray-tracing simulator and environmental data from OpenStreetMap (OSM), allowing realistic modeling of urban building layouts and propagation conditions across more than 40 global cities. The dataset includes radio channel responses under five carrier frequencies (2.6 GHz, 6 GHz, 28 GHz, 60 GHz, and 100 GHz), supporting a wide range of communication scenarios from sub-6 GHz to millimeter-wave (mmWave) applications.

Two primary scenarios are included in *WAIR-D*:

- **Scenario 1:** Comprising 10,000 unique environments, each containing 5 base stations (BSs) and 30 sparsely deployed user equipments (UEs), generating 150 radio links per environment. This scenario supports generalization studies across diverse environments.
- **Scenario 2:** Featuring 100 dense environments selected from Scenario 1, each with 1 BS and 10,000 densely placed UEs. It enables fine-tuning and transfer learning tasks under dense deployment settings.

The dataset structure is comprehensive and modular. Each environment includes files with propagation path data (e.g., delay, angle of arrival/departure), environmental metadata (BS and UE positions, region size), and simulated channel responses per frequency. Additional scripts and example tasks such as environment reconstruction and spatial beam prediction are provided to demonstrate dataset usage. Notably, the dataset allows user-specific channel synthesis, facilitating experiments with different antenna arrays, numerologies, and OFDM configurations.

The *Panoradio SDR* Dataset: HF Signal Classification with Simulated Impairments

The *Panoradio SDR* [4] dataset offers a collection of 18 waveform types primarily used in the high-frequency (HF) band (3–30 MHz), generated to support machine learning-based radio signal classification. Each signal is represented by 2048 complex I/Q samples at a sampling rate of 6 kHz, capturing a duration of approximately 340 ms. The dataset includes typical communication impairments such as additive white Gaussian noise (AWGN), Watterson fading (to emulate ionospheric propagation), and random frequency and phase offsets.

Signals in this dataset are modulated using audio-based sources, including speech, music, and text.

1.1.2. Technology-Specific RF Datasets

Beyond general-purpose RF signal datasets, there exists a broad set of resources tailored for highly specific applications, technologies, or frequency bands. These datasets often serve well-defined research objectives. While they contribute valuable insights within their domains, their focus limits their applicability in more general signal processing or machine learning frameworks aimed at broader wireless environments.

Nextly, several of these application-specific datasets are briefly reviewed and their specific orientation are highlighted.

WiSig: RF Fingerprinting of WiFi Devices

The WiSig dataset[5] contains signal captures from 174 off-the-shelf WiFi transmitters recorded across 41 USRP receivers. It was created to support research in RF fingerprinting, where the objective is to uniquely identify devices based on physical-layer imperfections in their transmissions. While rich in WiFi-specific signal diversity, the dataset is limited to a single technology and a specific application—device authentication—making it unsuitable for tasks involving modulation diversity or cross-technology interference.

DeepMIMO: mmWave and Massive MIMO Modeling

DeepMIMO [6] is a synthetic dataset based on ray-tracing simulations designed for mmWave and massive MIMO scenarios. It supports research in beamforming, channel estimation, and user localization at frequencies up to 60 GHz. However, the dataset is entirely synthetic and focuses solely on high-frequency scenarios with directional beam management, which diverges from the objectives of studies centered on interference-rich, heterogeneous wireless environments.

OPERAnet: Multimodal Activity Recognition

OPERAnet [7] provides multimodal sensor data—including WiFi CSI, passive radar, UWB, and RGB/depth images—from indoor environments to support activity recognition tasks. While it demonstrates the integration of RF and vision-based sensing, the dataset is geared towards spatial and behavioral inference rather than communication-oriented signal processing, and it lacks signal diversity across wireless technologies or modulation formats.

Bluetooth and WiFi Dataset for RF Fingerprinting

This dataset captures real-world Bluetooth and WiFi transmissions from commercial devices to support RF fingerprinting studies. The signals were recorded over several days and under variable conditions to reflect device-specific variations. Its focus on device

identification via physical-layer characteristics constrains its use to scenarios where transmitter identification is the primary goal, rather than signal classification, modulation analysis, or interference mitigation [8].

1.1.3. Interference Datasets

This subsection provides an overview of key publicly available interference datasets, detailing their structure, signal types, application domains, and limitations. Designed for tasks like protocol classification, jamming detection, or technology coexistence studies, these datasets focus on specific environments such as radar, WLAN, or Bluetooth. While useful for targeted use cases, they are not suited for developing general-purpose machine learning models for wireless interference recovery [9].

Radar Interference Dataset

One of the few public datasets designed for RF interference suppression is the Radar Interference Dataset by Ristea et al. [10]. This dataset was generated to support deep learning models in identifying and mitigating interference in automotive radar systems, a domain where accurate perception is critical and where interference can originate from neighboring vehicles operating similar radar systems.

The dataset comprises 48,000 samples, each corresponding to a simulated automotive radar signal. It contains three categories: interference-free signals, signals with interference, and a label vector indicating the complex amplitude values at the target location. While fixed parameters such as bandwidth, sampling frequency, center frequency, and sweep time were maintained across the dataset, other features like signal-to-noise ratio (SNR), signal-to-interference ratio (SIR), target amplitude, distance, and phase were randomly varied using uniform distributions.

Given the complex and dynamic nature of radar signals—exacerbated by the extensive spectrum sharing between radar and communication systems—accurately simulating such interference is challenging, which makes this dataset specially interesting. However, its scope remains quite limited. It includes only a single interference source per sample and focuses exclusively on FMCW automotive radar systems.

Radio Frequency Interference Dataset

The Radio Frequency Interference (RFI) Dataset presented by Ujan et al. [11] focuses on real-time satellite-to-ground communication signals combined with multiple jamming interferences. The dataset was generated by transmitting a digital video stream based on the DVB-S2 standard using a Software Defined Radio (SDR) setup involving GNU Radio and USRP hardware.

Three well-known types of jammers are incorporated into the dataset to simulate realistic interference scenarios: Continuous Wave Interference (CWI), Multi-Continuous Wave Interference (MCWI), and Chirp Interference (CI). These jamming signals are combined with the signal of interest (SoI) at varying signal-to-noise ratios to increase the dataset's complexity and relevance.

A distinctive feature of this dataset is the use of scalograms — the squared magnitude of the continuous wavelet transform (CWT) — to represent received signals in the time-frequency domain. These scalograms serve as inputs to several pretrained convolutional neural networks (CNNs) such as AlexNet, VGG16, GoogleNet, and ResNet18, facilitating direct feature extraction for interference recognition and automatic modulation classification (AMC).

The dataset comprises 4800 observations with multiple modulation types (QPSK, 8APSK, 16APSK, and 32APSK) and varying interference conditions.

CRAWDAD Dataset

The Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD) [12] was created to fill the gap in capturing real-world wireless network data, enabling a deeper understanding of how users and devices interact in diverse environments. As of the latest update, CRAWDAD hosts over 125 datasets covering various wireless communication applications, contributed by more than 14,000 users across over 120 countries.

A notable dataset within CRAWDAD, contributed by Schmidt et al. [13], comprises packet-level traces of IEEE 802.11b/g, IEEE 802.15.4, and Bluetooth transmissions with varying signal-to-noise ratios (SNR) in the baseband. This dataset includes fifteen different device classes, featuring ten IEEE 802.15.1 devices, three IEEE 802.11 devices, and two IEEE 802.15.4 devices. Additional datasets within CRAWDAD focus on HTTP traffic measurements over 802.11 in dense wireless classrooms and RSSI traces collected under controlled noise injections [14], [15].

Real-Time Interference Identification for IoT Devices

Grimaldi et al. [16] collected experimental data in four different environments, including interference-free, controlled interference, and uncontrolled real-world interference scenarios. The dataset comprises over 100,000 labeled interference bursts originating from Bluetooth and Bluetooth Low Energy (IEEE 802.15.1 and 802.15.1 BLE), ZigBee (IEEE 802.15.4), and WLAN (IEEE 802.11) devices.

A key contribution of this work is the heterogeneous nature of the interference, which supports the development and evaluation of interference identification methods suitable for diverse wireless coexistence scenarios.

1.1.4. Applications of Wireless Signal Datasets: Use Cases

Public wireless signal datasets play a crucial role in advancing machine learning techniques for signal processing tasks such as interference rejection, modulation classification, and signal recovery. These datasets provide standardized benchmarks that enable researchers to develop, compare, and refine algorithms under realistic and challenging conditions.

The RF Challenge

Building on the earlier discussion of the *RF Challenge* dataset [17], the RF Challenge [18] further explores the issue of single-channel RF signal separation in the presence of overlapping interference from diverse wireless technologies such as WiFi, Bluetooth, ZigBee, and 5G. This dataset includes mixtures where interference overlaps with the signal of interest in both time and frequency, often with minimal or no prior knowledge of the interference's generation process, making it a challenging real-world scenario for traditional methods.

The RF Challenge has significantly advanced data-driven interference mitigation techniques, with teams developing deep learning models like UNet and WaveNet architectures. These models outperform classical methods by orders of magnitude in bit error rate and mean squared error, setting a new benchmark for testing interference rejection approaches and pushing the development of solutions that can generalize across various interference types.

The RF Challenge dataset [1] was designed to replicate realistic RF interference conditions, featuring four signal mixtures that combine up to four wireless technologies, including WiFi, Bluetooth, ZigBee, and 5G. Captured using software-defined radios, these mixtures reflect real-world channel effects and hardware imperfections. Presented as single-channel complex time-domain signals without source labels, the task is framed as blind source separation. Its realism and diversity make it a key benchmark for evaluating deep learning approaches in RF signal recovery.

The Automotive Radar Use Case

Another illustrative example of how publicly available datasets are employed in wireless signal processing is found in the domain of automotive radar systems. Ristea et al. [10], as discussed in Oyedare's thesis [9], developed a dataset of FMCW radar signals annotated for interference presence and target location. While previously described in Section 1.1.3, its relevance here lies in its use for training supervised deep learning models capable of identifying interference and extracting spatial features from radar returns.

1.2. Motivation and Relevance of the Study

The primary motivation for this work arises from the growing need for high-quality, large-scale datasets specifically tailored for wireless communication signal processing. While fields like image and audio processing have an abundance of publicly available datasets, the same cannot be said for radio frequency (RF) signals. This lack of structured and versatile datasets limits the development of robust machine learning models for applications such as signal classification, interference rejection, and signal recovery in wireless environments. Existing datasets are often narrow in scope, limited in diversity, or tied to very specific use cases or technologies.

A review of current datasets highlights these gaps clearly. The *RF Challenge* dataset is valuable for signal separation and interference mitigation in RF environments, but it has limitations as a general benchmark. Many signals, such as *EMISignal1*, *CommSignal2*, and *CommSignal3*, lack detailed information on their generation process, limiting their use for models that rely on understanding signal structures. Additionally, the dataset is focused on specific RF scenarios and does not include a variety of wireless technologies or higher-layer protocol behavior, making it more suited for interference testing rather than general wireless communication tasks [18].

The widely used *RadioML 2018.01A* dataset provides a rich variety of modulations under controlled impairments, including an over-the-air (OTA) component, but it lacks cross-technology interference scenarios and is based solely on randomly generated bitstreams without application-level structure or temporal coherence. Additionally, it is centered around basic modulation formats such as PSK and QAM, without targeting specific wireless technologies like Wi-Fi, Bluetooth, or 5G. As a result, the dataset does not incorporate higher-layer protocol behavior or realistic network-level traffic structures that would be encountered in practical wireless communication systems [2].

WAIR-D, while comprehensive in its modeling of radio channels and environmental realism, focuses exclusively on channel-level information and does not include different modulated baseband signals or structured traffic patterns relevant to signal classification tasks. This makes this dataset solely focused on the study of propagation characteristics and environment-aware modeling, rather than on the analysis of modulated signals or communication protocols [3].

Similarly, the dataset introduced by Scholl [4] focuses on HF radio transmissions using synthetic signals based on modulated audio sources, such as speech or music, which may lack the consistent temporal structures or complexity found in higher-layer application traffic like video streams. Moreover, its scope is restricted to the high-frequency (HF) band, which, while relevant for long-range communication and amateur radio, is not representative of modern wireless systems that operate predominantly in the UHF and microwave ranges, such as Wi-Fi, LTE, or 5G [4].

Other datasets are highly specific in their design: *WiSig* is devoted to RF fingerprinting of WiFi devices, offering signal diversity across transmitters but only within a single

technology and task [5]. *DeepMIMO* is synthetic and focused on mmWave and massive MIMO modeling for beamforming and localization applications [6]. *OPERAnet* combines RF and vision data for activity recognition, emphasizing behavioral inference rather than signal analysis [7]. Similarly, the dataset from Jagannath et al. captures real-world WiFi and Bluetooth transmissions for fingerprinting, but focuses on prolonged emissions and lacks the signal diversity and traffic structure required for broader signal processing studies [8].

Finally, regarding datasets focused specifically on interference, the *Radar Interference Dataset* by Ristea et al. [10] focuses exclusively on FMCW automotive radar signals with a single interference source under controlled simulation parameters. While valuable for radar interference studies, it lacks diversity in signal types and interference scenarios. Also, dataset's controlled conditions and lack of signal diversity make it unsuitable for training general-purpose foundation models intended to operate across a wide range of wireless scenarios.

The *Radio Frequency Interference* (RFI) Dataset by Ujan et al. [11] includes satellite communication signals with several jamming types, represented as scalograms for deep learning. However, its scope is limited to a specific satellite communication setting and predefined jammers, without support for flexible modulation schemes or layered traffic structures.

The *CRAWDAD* repository [12], [13] provides extensive real-world traces from Wi-Fi, Bluetooth, and IEEE 802.15.4 devices. Although rich in real-world diversity, CRAWDAD datasets primarily consist of packet-level captures lacking detailed physical-layer impairments or controlled interference injection. This limits their use for training models that require fine-grained signal degradation knowledge.

Grimaldi et al. [16] gathered over 100,000 interference bursts from heterogeneous wireless devices in various environments to enable real-time interference identification on IoT hardware. While providing valuable insights into interference classification, their dataset focuses on specific 2.4 GHz band technologies and hardware constraints, limiting generalization.

In light of the limitations observed in existing datasets and the inherent complexity and diversity of modern wireless communication scenarios, this research proposes the creation of a dataset designed not around a single technology or fixed interference conditions, but to reflect the broad range of modulations, impairments, and traffic structures found in real-world systems. While the dataset itself does not include interference signals directly, it is designed with modularity and extensibility in mind, providing flexible tools to easily incorporate synthetic and real interference, as well as to customize impairments and signal conditions. This approach, combined with structured traffic patterns derived from application-level sources such as video streams, aims to support a wide variety of machine learning applications in signal recovery, classification, interference mitigation or any other signal processing technique applicable. Additionally, unlike in challenges such

as the *RF Challenge* [18], where no prior information about the signals is provided, our framework allows full access to metadata and signal properties, facilitating reproducibility, interpretability, and debugging. Ultimately, this work seeks to lay a more realistic, robust, and reproducible foundation for data-driven wireless communications research, empowering users to tailor datasets and scenarios to their specific needs.

1.3. Contributions and Structure of the Thesis

The main contribution of this thesis is the generation of a diverse and high-quality dataset for wireless communication signal processing. This dataset includes several different modulation techniques and configurations and has been designed to support machine learning applications, specifically for tasks like signal classification, interference detection, and general signal processing. Unlike other datasets based on random bitstreams, the signals in this dataset are generated from structured video data, which introduces multiple layers of complexity and temporal coherence. This additional structure provides a more realistic representation of real-world communication scenarios, allowing for more detailed evaluation and revision. Importantly, the dataset will be released as a free and open-access resource, enabling the research community to use, reproduce, and extend the experiments presented in this work.

To facilitate the generation of this dataset, several applications and tools have been developed in both MATLAB and Python languages, complementing each other. This applications will be provided and explained throughout this project.

- **MATLAB Tools for Signal Generation and Visualization of Interferences:** A set of MATLAB applications was developed to generate wireless signals with various modulation schemes and noise conditions. These tools also allow for the comparison of multiple signals simultaneously, facilitating the analysis of different signal types and interference scenarios.
- **Python notebook and scripts for Signal Visualization and Interference Generation:** A Python-based framework has been developed to visualize the signals generated in MATLAB. This includes visualizing time-domain signals, spectrograms, constellations, and signal power distributions. Additionally, some other Python scripts have been developed with functions for generating new datasets containing interfered signals by combining and manipulating the original clean datasets among themselves. These capabilities offer insights into signal characteristics and the impact of noise and interference, helping to assess and extend the dataset's suitability for diverse machine learning tasks.

These contributions provide a solid foundation for future research in machine learning for wireless communication. The dataset and associated tools enable researchers to explore new methods for signal processing using machine learning techniques. Moreover, the framework developed throughout this thesis serves as a comprehensive, data-driven solution for wireless communications. It is intuitive and easy to use across a wide range

of applications, particularly in the realm of wireless signal processing. This framework not only includes the tools for dataset generation and visualization, but also provides the necessary utilities to evaluate the performance of machine learning models in real-world tasks, such as interference mitigation.

This work demonstrates the use of the entire system to evaluate interference mitigation models. The dataset generation tools, along with signal visualization and manipulation capabilities, are employed to assess model performance. This integrated framework highlights the practical and scalable potential of machine learning for signal processing in wireless communications.

1.3.1. Structure of the Thesis

This document is organized into four main chapters, each addressing a fundamental part of the project. The first chapter serves as an introduction to the overall context and goals of the thesis. It begins by outlining the motivation behind the work and reviewing the current state of the art in wireless communication datasets, highlighting their structures, signal types and limitations. This chapter also explores some of the main applications of these datasets. It concludes with a brief overview of the proposed dataset and a description of the software tools and environments used throughout the project.

The second chapter focuses on the generation and organization of the dataset. It explains the types of signals included, the rationale behind their selection, and how they were generated using MATLAB tools and application-level traffic sources such as video streams. It also describes the structure and format of the dataset files, stored using HDF5, and the additional tools created in Python to assist with visualization and signal interference. Particular attention is given to the development of a modular structure that allows for the future addition of other signals or impairments in a controlled and reproducible way.

In the third chapter, a case study is presented to demonstrate the usefulness and flexibility of the dataset. A UNet-based neural network is trained using the generated signals, and its performance in recovering signals under synthetic interference conditions is evaluated. The chapter includes both quantitative metrics and visual comparisons using MATLAB applications to assess the effectiveness of the interference mitigation process.

Finally, the fourth chapter summarizes the results and conclusions of the project. It highlights the effectiveness and flexibility of the developed dataset, emphasizing its ability to support a wide range of wireless signal processing tasks. The chapter also outlines several promising future directions for extending the framework, such as incorporating additional wireless technologies, expanding the dataset's diversity, and enhancing machine learning models for more robust signal recovery and interference mitigation. The potential of this dataset and framework as a foundation for advanced research in data-driven wireless signal processing is thoroughly discussed, opening pathways for further

exploration in the field.

1.4. Tools and Software Used

This section presents the main tools and software environments used throughout the development of this project, both for signal generation and processing, as well as for dataset construction and machine learning experimentation.

1.4.1. MATLAB

MATLAB has been the primary environment used for signal generation, signal processing, and data preparation. Leveraging the Wireless Waveform Generator toolbox [19], custom scripts were developed to synthesize realistic wireless communication signals, ranging from basic signals such as PSK or QAM, even OFDM, to more complex signals including WiFi or Bluetooth transmissions. These scripts allow flexible configuration of modulation schemes and the injection of AWGN.

In addition, MATLAB was used for demodulating and decoding received signals, enabling a clear evaluation of signal recovery performance. It was also instrumental in preparing visual comparisons between clean, interfered, and recovered video streams, helping validate the qualitative impact of interference suppression.

Custom MATLAB functions were also developed to facilitate dataset generation in HDF5 format, to ensure proper organization of signal metadata and consistency across training samples.

1.4.2. Python

Python was used primarily for signal visualization and dataset augmentation. A dedicated Jupyter notebook framework was built to visualize time-domain waveforms, frequency-domain characteristics (via spectrograms), constellation diagrams, and signal energy distributions. This visualization suite allowed for fast qualitative assessment of the impact of impairments and interference.

Additionally, the Python framework includes utilities for generating new datasets by combining clean signals with synthetic or captured interference, enabling controlled experimentation. These tools ensure reproducibility of experiments and flexibility in scenario construction for machine learning tasks.

The development and execution of Python code were carried out mainly using Google Colab and Visual Studio Code, providing an efficient and flexible environment for experimentation, visualization, and collaboration.

1.4.3. Computational Resources

The training of deep learning models for signal recovery was conducted using an NVIDIA RTX 6000 Ada Generation GPU, which provided the necessary performance for handling large datasets and complex architectures. This machine is part of the computational infrastructure of the Signal Theory Department, and its use was carried out under the supervision of Dr. Alejandro Lancho, the advisor of this thesis. All signal generation and dataset preparation tasks were carried out on a local workstation (HP 15s laptop), which was sufficient for running MATLAB and Python tools efficiently. This separation of computational workloads allowed for a streamlined workflow from signal synthesis to neural network evaluation.

2. Dataset Generation and Description

As already introduced, the main objective of this project is to generate a sufficiently large dataset of wireless communication signals that can serve as a reference for training future models in signal processing.

This dataset will be designed to provide all the necessary resources for working with wireless signals, covering the entire process from signal generation to the proper evaluation of processing techniques applied to them. To achieve this, the dataset will include not only the waveform representations of the signals but also the original bit sequences used for their generation.

2.1. Overview of Wireless Signals and Datasets

Wireless communication systems rely on the transmission and reception of radio frequency (RF) signals to convey information over the air. These signals are typically represented in their baseband form using in-phase (I) and quadrature (Q) components, which together constitute the complex envelope of the waveform. The I/Q representation provides a complete description of the signal's amplitude and phase variations, which are essential for both modulation and demodulation processes. In practical digital implementations, these I and Q components are sampled at discrete time intervals, resulting in a sequence of complex-valued samples—each containing a real (I) and an imaginary (Q) part—that represent the time-domain waveform of the signal. These complex samples are the standard input for signal processing algorithms and machine learning models, as they encapsulate all the dynamic information required to reconstruct and analyze the signal behavior. Therefore, these complex-valued sequences of samples will be the components of the waveform of our datasets [20].

Modulation is a fundamental process in wireless systems, whereby digital information (bitstreams) is encoded onto an analog carrier wave by varying one or more of its properties—typically amplitude, frequency, or phase. Common modulation schemes include phase-shift keying (PSK), quadrature amplitude modulation (QAM), and orthogonal frequency-division multiplexing (OFDM), each offering different trade-offs in terms of bandwidth efficiency, resilience to noise, and complexity. Some of them will be studied more deeply throughout this projects.

These modulated baseband signals are then upconverted to a carrier frequency suitable for transmission over a specific wireless channel. The RF spectrum is typically divided into different bands, such as the 2.4 GHz and 5 GHz industrial, scientific, and medical (ISM) bands used by technologies like Wi-Fi and Bluetooth, or the sub-6 GHz and millimeter-wave (mmWave) bands used in LTE and 5G systems.

The received signals are usually corrupted by various channel impairments, including additive white Gaussian noise (AWGN), fading, Doppler shift, and interference from other

users or technologies. The I/Q signal format allows these effects to be captured faithfully, making it an ideal representation for processing and machine learning applications.

Concerning signal processing tasks based on ML/AI techniques, access to large collections of labeled I/Q samples (organized in well-structured datasets) is essential. Such datasets must not only include the raw signal waveforms, but also metadata such as the modulation type, signal-to-noise ratio (SNR), interference characteristics, and the original bit sequences, to enable proper training and evaluation of these machine learning models.

2.1.1. Signal Types to be Covered in the Dataset

The dataset developed in this work is designed to encompass a broad spectrum of wireless communication signal types, ranging from simple, well-understood modulation schemes to complex real-world communication standards. Initially, the dataset will include baseband waveforms generated using canonical modulation formats such as phase-shift keying (PSK), quadrature amplitude modulation (QAM), and orthogonal frequency-division multiplexing (OFDM). These fundamental modulations serve as a controlled baseline for evaluating signal processing and machine learning algorithms.

Beyond these, it will also incorporate more sophisticated wireless communication signals. Specifically, it will cover signals operating in the ISM radio bands, such as Bluetooth and several Wi-Fi standards. These signals present a higher level of complexity due to their protocol stack structure, time-varying traffic patterns, and coexistence behavior in dense spectrum environments.

Canonical Modulation Schemes

Canonical digital modulation schemes form the foundation of most wireless communication systems. In this dataset, we include a range of commonly used modulations to provide a controlled environment for evaluating machine learning models and signal processing algorithms.

Phase-Shift Keying (PSK) is a family of modulation schemes where the phase of the carrier wave is varied according to the digital information. The dataset includes binary PSK (BPSK), quadrature PSK (QPSK), and 8-PSK to represent low- to moderate-complexity systems [20].

Quadrature Amplitude Modulation (QAM) combines both amplitude and phase variations, allowing for higher spectral efficiency. The dataset includes widely used variants, up to 4096-QAM, which are prevalent in standards like Wi-Fi and LTE [20].

Orthogonal Frequency-Division Multiplexing (OFDM) is a multicarrier modulation scheme that divides the signal bandwidth into multiple orthogonal subcarriers, each modulated independently. OFDM is used in nearly all modern wireless systems due to its robustness against multipath fading and frequency-selective channels. The dataset

includes pure OFDM signals modulated using QAM on each subcarrier, enabling experiments on time-frequency domain behavior [20].

Direct Sequence Spread Spectrum (DSSS) is a spread-spectrum technique in which the transmitted signal is spread over a wider frequency band than the original signal bandwidth using a pseudo-random code. DSSS improves resistance to narrowband interference and enhances signal security and reliability. It is employed in technologies such as IEEE 802.11b and early versions of Bluetooth. Including DSSS signals in the dataset enables the study of spreading effects and their interactions with various types of interference and noise [21].

These canonical modulations are generated in isolation and they form the baseline for more complex signal environments addressed later in the project.

Technology-Specific Communication Signals: Bluetooth and Wi-Fi

In addition to basic modulation formats, the dataset also includes waveforms extracted from real-world communication technologies that operate in both the ISM bands and licensed cellular spectrum. These signals reflect the complexity and diversity of practical communication environments, where higher-layer protocol structures, bursty traffic patterns, and technology coexistence play a crucial role.

Bluetooth signals, primarily operating in the 2.4 GHz ISM band, are characterized by short-range frequency-hopping spread spectrum (FHSS) transmissions. The dataset includes both classic Bluetooth (BR/EDR) and Bluetooth Low Energy (BLE) modes, which differ in their data rates, access methods, and channel usage [22].

Wi-Fi signals are generated using various versions of the IEEE 802.11 standard, including 802.11ac [23], 802.11n [24], 802.11b [23] and 802.11ax (HE) [25]. These signals incorporate complex MAC-layer structures and exhibit dynamic behavior based on user activity, channel conditions, and quality of service mechanisms.

These technology-specific waveforms were selected to replicate real-world multi-standard wireless environments. By including such diverse sources, the dataset supports experimentation with cross-technology interference, multi-user detection, and signal classification under realistic traffic patterns.

2.1.2. Role and Limitations of Existing RF Datasets

Datasets play a fundamental role in the development of AI and machine learning applications for RF signal processing. As discussed previously, large-scale and well-annotated datasets are essential for training, testing, and benchmarking models capable of operating in complex wireless environments.

Classification of RF Signal Datasets

As we have already discussed, several training dataset for RF signals have been developed and published lately with different purposes and, therefore, different characteristics. According to *Panoradio SDR* [26], some key differentiating factors are the following:

- **Data classes or labels**: The specific set of modulation types or signal categories included.
- **Source of RF data**: Whether signals are generated via software simulation, recorded in controlled laboratory environments, or captured from real-world wireless systems.
- **Augmentation**: The extent and nature of signal impairments introduced, such as noise, fading, and frequency offsets.
- **Data format**: Representation of the signals, e.g., I/Q samples versus spectral features, sampling rates, and observation duration.
- **Number of data samples**: The overall size and diversity of the dataset.
- **Quality of data and labels**: Accuracy of labeling, presence of noisy or corrupted samples, and dataset biases.

2.1.3. Proposed Dataset

Attending to this classification, we could say that our dataset will be a fully synthetic dataset generated using MATLAB's Wireless Waveform Generator app. It will cover a variety of modulation types and wireless technologies, with signals generated by modulating structured video data instead of random bitstreams. For each waveform, the dataset provides two sequences representing the real (in-phase) and imaginary (quadrature) components of the complex baseband samples (I/Q samples) along with the modulated bits (from the video). The number of bits modulated per signal is kept constant, so the number of samples for each modulation or technology in the dataset will vary, but always representing the exact same bitstream. Although the dataset itself does not include impairments or interference, it is designed with flexibility to easily incorporate such effects, supported by dedicated tools. The dataset also specifies different SNR values for each signal as controlled by the waveform generator, facilitating comprehensive evaluation scenarios.

2.2. Signal Generation from Videos

A key advantage of our dataset generation process is the use of structured bitstreams derived from real video files in *.mp4* format, rather than conventional randomly generated bits. By modulating these actual video bitstreams, the resulting wireless signals encapsulate meaningful data that should be recoverable from the received signals. This

approach provides an additional layer of validation: beyond traditional analytical metrics, the quality of signal recovery can be assessed visually by reconstructing the original video content, offering a more intuitive and comprehensive evaluation of the processing algorithms' effectiveness.

Also, the bits extracted from real videos maintain an inherent structure that reflects the temporal and spatial dynamics of the video content. This structure is not only relevant for signal modulation, but it also provides an additional layer of information that machine learning models can leverage. This allows the model to not only base its understanding on the modulation but also on the temporal and spatial relationships that the video-derived bits inherently contain, improving its performance in signal recovery and reconstruction tasks.

Coherent Length of the Signals in the Dataset An important point to take into account is that every video format of common use is based on a dynamic compression method, meaning that the length of the video depends on the video content, not only on the number or size of the frames. This makes these video formats unideal for the generation of a dataset, where it is wanted for all signals to be the same length. With this in mind, the bits to be modulated for the dataset will be the raw bits from the video, which, by using the same number of frames and frame size, will generate equally long videos.

Video Source The video content primarily consists of stock footage obtained from *Vecteezy* [27], a publicly accessible online platform offering free videos for download. These videos cover a variety of scenes and motion complexities, ensuring a diverse range of bitstream structures that enrich the dataset and challenge machine learning models to generalize over realistic, temporally correlated data.

2.3. MATLAB application for Dataset Generation

The tool for the generation of these datasets has been developed in MATLAB as a graphic user interface, allowing plenty of freedom for the user to select different configurations of the datasets.

The tool is accessed by calling the function `DatasetGeneration.m`, which will open a window to select the folder with the videos (that must be in *.mp4*, even though the implementation of other formats in the future should not be a problem). Once this folder is selected, a new window will appear, where the user must choose the videos that will be used for generating the signals. These videos will be divided into smaller videos of several frames so that the waveforms are easier to work with. Therefore, in addition to choosing the videos, the user must also decide the number of videos into which the original video will be divided and the number of frames that each *subvideo* will have. It is interesting to mention that, as all videos are processed in the same way at the same time, the total number of frames to be used (number of *subvideos* x number of frames per video) must

not be larger than the total number of frames of the video with the least number of frames from the selected ones. This value is displayed right below the boxes to complete with the number of *subvideos* and frames per video for the user to take into account. There is also an option to select a random number of videos.

In addition, the original videos can be previewed in this window (shown in Figure 2.1).

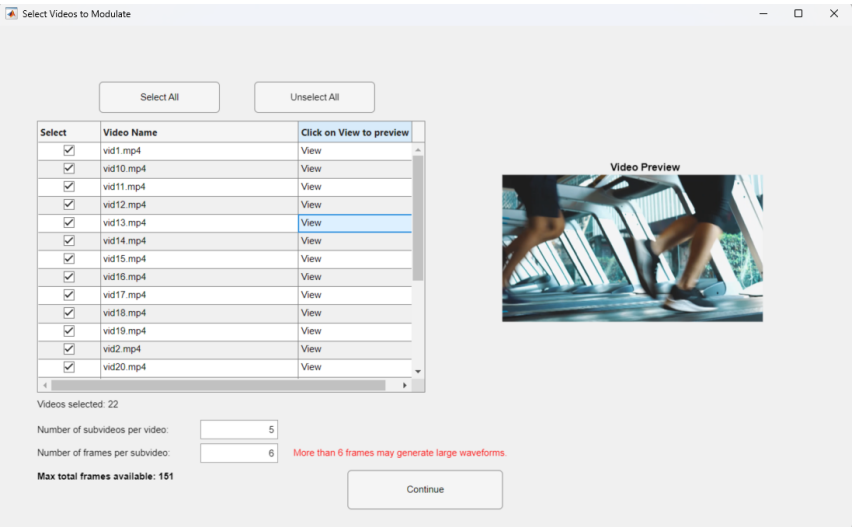


Fig. 2.1. Window to select the videos to modulate

Once this initial configuration is set, the *Continue* button must be clicked. This will open a new window displaying the list of the new videos to be modulated. Just like in the previous window, these videos can also be previewed. These are not simply the *subvideos* with the previously defined number of frames, but also a processed version of them. The videos are converted to black and white, reduced in quality and cropped to a uniform frame size of 50×50 pixels (padded with black frames if needed). This ensures that all videos have the same length and makes them lighter (for any future use with machine learning algorithms or any other processing) while keeping the video images and motion. This also eliminates the need for compression, which would otherwise alter their duration and create inconsistencies in the dataset. The bits used for modulation are extracted directly from these processed videos (Figure 2.2).

Again, after clicking on the *Continue* button, a new window will appear showing the different modulations we can apply. For every available modulation, a set of parameters must be configured, as shown in Figures 2.3a and 2.3b, where an example for the OFDM modulation is selected.

It is worth mentioning that several different modulation configurations can be selected. This means, the user can select several modulations but also the same modulation several times but with different configurations. In case the same modulations with the same parameters are selected, an error will appear and will force the user to select another configuration.

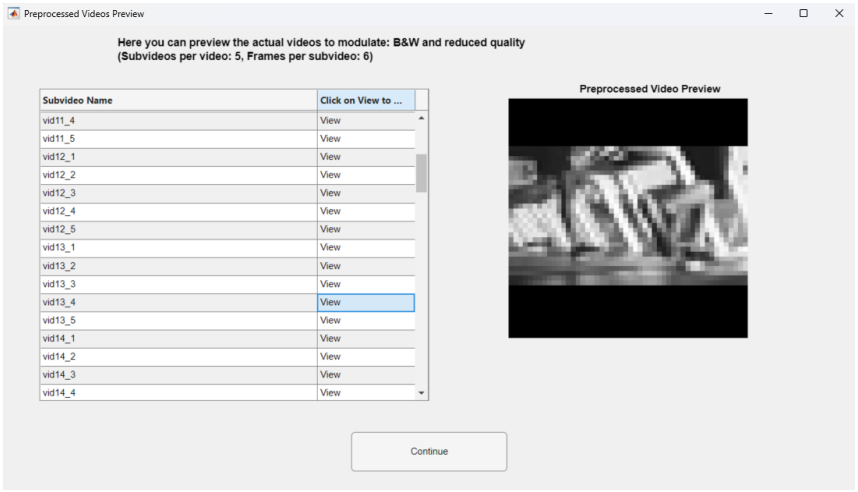
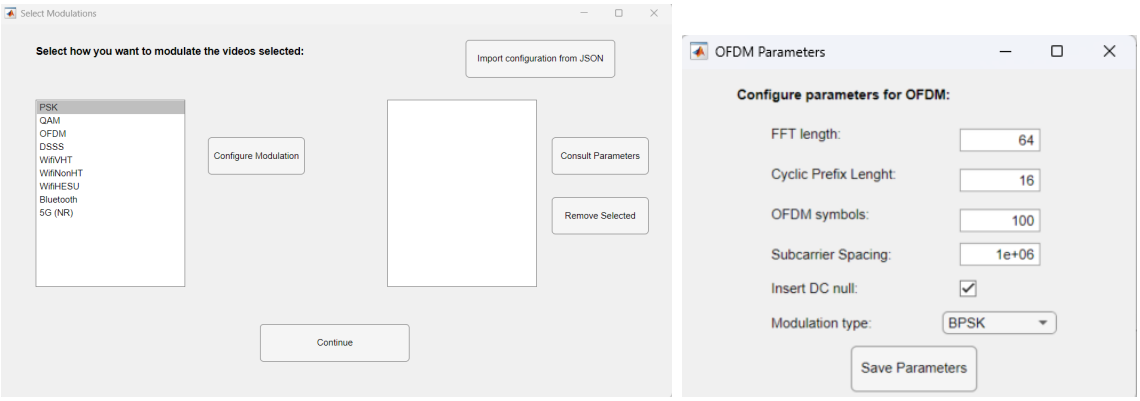


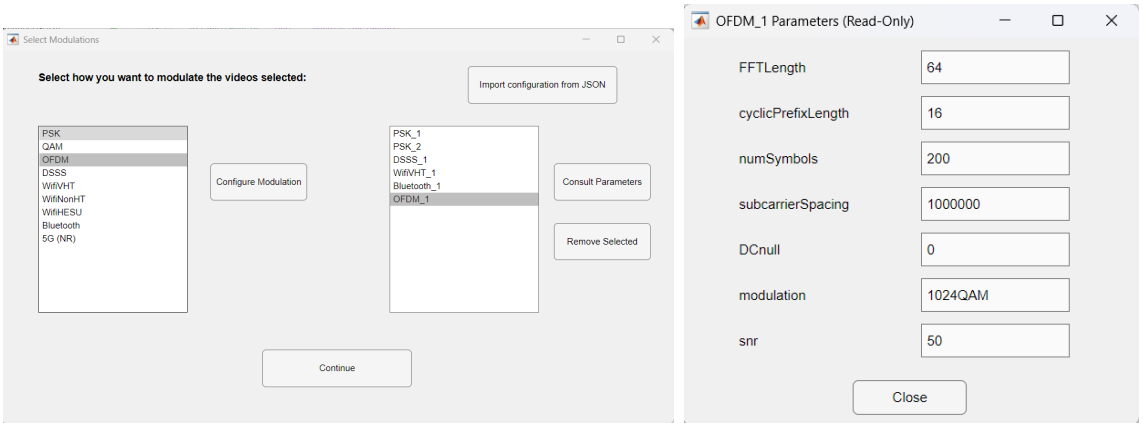
Fig. 2.2. Window to preview the actual videos to be modulated



(a) Selection of modulations (b) Configuration of OFDM modulation

Fig. 2.3. Modulation selection window

As the user adds these modulations (or configurations), these will be appearing in the box at the right part, where these configurations can be selected for consulting their parameters or for eliminating. In case the user wants to alter some (or all) of the parameters of one configuration already saved, the user should eliminate it, select it and save it again. All these can be seen in Figures 2.4a and 2.4b.



(a) Selected and saved modulations (b) Consultation of OFDM parameters

Fig. 2.4. Selected and consulted configuration windows

It is also worth mentioning the *Import configuration from JSON* feature in this window. As will be explained later, in addition to saving the datasets containing the signals, the configurations used for modulation and other relevant parameters are stored in a JSON file located in the same folder. With this in mind, it was considered a valuable enhancement to allow users to reuse these configurations. This feature can save time when generating a new dataset with the same configuration, but with different signals or lengths, or simply when it is necessary to repeat the dataset creation process.

Finally, after clicking on the *Continue* button, the user will be redirected to the final window. In this one (Figure 2.5) a list of the selected modulations is displayed again and there is a box for the user to input the name of the folder where the dataset folder will be saved. By default, this creates a *datasets* folder in the parent directory of the path where the function `DatasetGeneration.m` is being run and a new folder with the input folder name will be created with all corresponding folders (in case the *datasets* folder already exists, no folder will be created and this one will be used). These will be explained lated. By default, the name of the new dataset folder is `dataset_yyy_mm_dd_n`, corresponding to the date and a cardinal number, in case several dataset are created on the same day.

When the *Start Modulation* button is pressed 2 progress bars will appear (Figure 2.6), one for the progress of the current modulation and one for the total progress of the dataset creation.

When all the modulations have been performed and saved, the window in Figure 2.7 will pop, indicating that the dataset has been generated successfully.

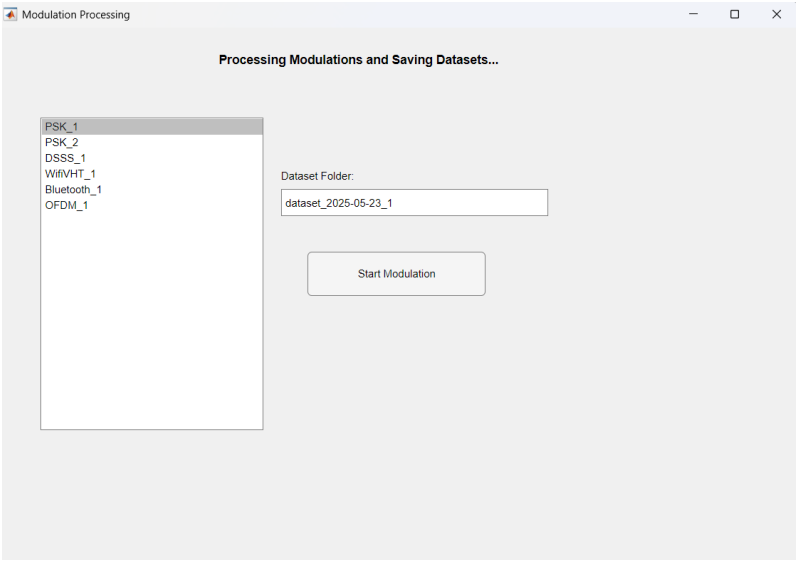


Fig. 2.5. Final window: Start modulation and saving dataset

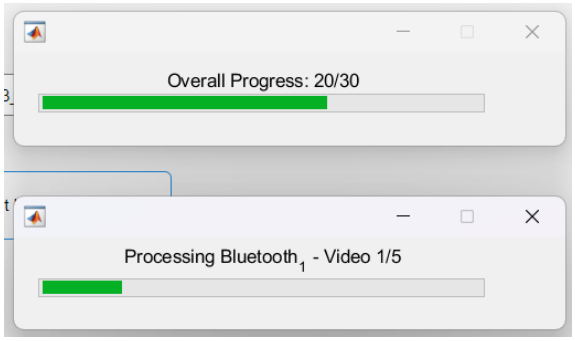


Fig. 2.6. Progress bars

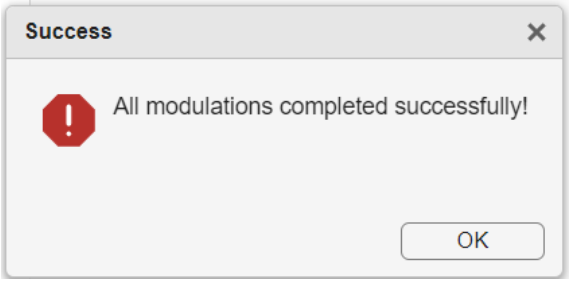


Fig. 2.7. Dataset generation completed successfully window

2.3.1. Types of Signals and Modulations

For each of the modulations, a MATLAB function has been created, where the inputs are the bits to be modulated (from the videos) along the parameters for each modulation (a struct with different fields depending on the modulation). All functions are based on the *WirelessWaveformGenerator* toolbox from MATLAB [19], with little modifications to allow longer bitstreams, so that instead of generating several waveforms for each bitstream, a longer waveform, product of the concatenation of these other waveforms, is generated. In the following subsections, the different modulations or technologies implemented, along with the parameters needed for each, will be explained.

PSK (Phase Shift Keying)

Phase Shift Keying (PSK) [20] is a digital modulation technique where the phase of a carrier signal is varied in proportion to the input data signal. In PSK, each symbol represents a specific phase shift, and the number of distinct phase shifts determines the order of the modulation. The more phase shifts there are, the more bits can be encoded per symbol, increasing the data rate. PSK is widely used in communication systems due to its relatively simple implementation and robustness to noise. Variants of PSK include Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK), and higher-order PSK schemes like 8PSK, 16PSK, and beyond, although in this work the higher PSK modulation to be used will be 8-PSK.

In the function we have implemented, the number of input bits must be a multiple of $\log(M)$, where M is the modulation order. If the number of bits provided does not meet this requirement, the excess bits are discarded, as seen in the functionality of the PSK modulation function in the toolbox [19].

The parameters needed for this modulation are:

- **Modulation Order**: Defines the number of distinct phase shifts used in the modulation scheme. For example, BPSK has 2 phase shifts, QPSK has 4, 8PSK has 8. The modulation order determines the number of bits per symbol. A higher modulation order allows more bits per symbol, which increases the data rate but also requires a higher signal-to-noise ratio (SNR) to achieve reliable communication.
- **Symbol Rate**: The rate at which symbols are transmitted. It represents the number of symbols transmitted per second (baud rate). The symbol rate affects the bandwidth of the communication system.

PSK modulation is a fundamental technique used in many communication systems, including as the base for more complex schemes like OFDM. It is widely used due to its simplicity and efficiency. It is also one of the basic modulation techniques found in various datasets mentioned in the State of the Art 1.1, which is why we considered it important to include it in our project.

QAM (Quadrature Amplitude Modulation)

Quadrature Amplitude Modulation (QAM) [20] is a digital modulation technique that combines both phase and amplitude modulation to increase the data rate. In QAM, each symbol represents a unique combination of amplitude and phase, allowing for the transmission of multiple bits per symbol. Similar to PSK, the modulation order determines the number of distinct symbols, with higher orders allowing more bits per symbol but requiring a higher signal-to-noise ratio (SNR) for reliable transmission.

In the function we implemented, the number of input bits must be a multiple of $\log(M)$, where M is the modulation order. Unlike PSK, which discards excess bits, QAM uses padding to handle bits that do not perfectly fit the symbol size. This ensures that no information is lost, and the original video length, stored in the modulation parameters archive, can be recovered by ignoring the padding bits.

The parameters needed for this modulation are the same as PSK:

- **Modulation Order**: Defines the number of distinct amplitude-phase combinations. For example, 16-QAM has 16 distinct symbols, 64-QAM has 64, and so on. The modulation order determines the number of bits per symbol.
- **Symbol Rate**: The rate at which symbols are transmitted, affecting the bandwidth of the system.

QAM is widely used in communication systems, such as WiFi and cellular networks, due to its ability to transmit large amounts of data efficiently. It is closely related to PSK and serves as the basis for more advanced modulation schemes like OFDM. Given its presence in various datasets mentioned in the State of the Art 1.1, again, it was essential to include QAM in our project.

OFDM (Orthogonal Frequency Division Multiplexing)

OFDM [20] is a digital modulation technique that splits a high-data-rate signal into multiple lower-rate subcarriers, which are orthogonal to each other. Each subcarrier is modulated individually using traditional schemes like BPSK, QPSK, or QAM. It is the foundation of many modern wireless standards, including WiFi (802.11a/g/n/ac/ax), 4G LTE, and 5G NR. Being the base of several standards makes this modulation an excellent basis for training future models, and therefore interesting to add to this tool.

The parameters needed for this modulation are:

- **FFT Length**: Defines the number of subcarriers used in the OFDM system.
- **Cyclic Prefix Length**: A portion of each OFDM symbol that is copied to its beginning to combat inter-symbol interference (ISI) caused by multipath propagation.
- **Number of Symbols**: Specifies the total number of OFDM symbols in a transmission. This determines the duration of the transmission block.

- **Subcarrier Spacing**: The frequency separation between adjacent subcarriers.
- **Insert DC Null**: A boolean parameter indicating whether a null subcarrier is inserted at the DC (center) frequency.
- **Modulation Type**: The digital modulation scheme applied to each subcarrier. It can be one of BPSK, QPSK, 16QAM, 64QAM, 256QAM, or 1024QAM.

If the *Insert DC Null* option is disabled, the input bitstream is processed in batches of 5300 bits. However, when *DC Null* is enabled, the bitstream is instead processed in batches of 5200 bits. Regardless of this setting, the resulting waveform generated from each batch will always consist of 8000 samples. This way, the final waveform will be a multiple of 8000 depending on the length of the input bits.

DSSS (Direct Sequence Spread Spectrum)

The second modulation implemented is Direct Sequence Spread Spectrum (DSSS) [21]. DSSS is a spread spectrum technique in which the input data is multiplied by a spreading code that increases the signal bandwidth before transmission. This spreading operation enhances resistance to interference, jamming, and multipath fading. DSSS is widely used in wireless communication systems, including the IEEE 802.11b WiFi standard, which, as in the previous case, makes it a good choice to add to the dataset.

The DSSS modulation scheme varies depending on the data rate selected. At lower rates, a Barker code of length 11 is used for spreading, while at higher rates, Complementary Code Keying (CCK) is applied. The modulated symbols are transmitted using DBPSK, DQPSK, or CCK, depending on the chosen rate.

The only parameter required for this modulation is the **Data Rate**. It determines the transmission rate and the corresponding modulation scheme. Supported values are 1 Mbps (DBPSK), 2 Mbps (DQPSK), 5.5 Mbps (CCK), and 11 Mbps (CCK).

This function is not implemented using the *WirelessWaveformGenerator* toolbox, but has been developed for this project. The modulation process is as follows:

- 1 Mbps and 2 Mbps: The input bits are mapped to BPSK (DBPSK) or QPSK (DQPSK) symbols. These symbols are then spread using a Barker sequence of length 11, resulting in an expanded signal.
- 5.5 Mbps and 11 Mbps: The input bits are directly mapped using CCK modulation, which does not require a separate spreading code.

The final transmitted waveform length depends on the spreading rate:

- For 1 Mbps, each input bit is mapped to 11 chips.
- For 2 Mbps, each input bit pair (2 bits) is mapped to 11 chips.
- For 5.5 Mbps and 11 Mbps, the chips are directly generated by the CCK encoder.

Wi-Fi HE-SU (802.11ax)

The High-Efficiency Single-User (HE-SU) mode of Wi-Fi 802.11ax [25] enhances spectral efficiency, increases data rates, and improves interference management. HE-SU uses Orthogonal Frequency Division Multiple Access (OFDMA), allowing multiple devices to share the same channel simultaneously, and supports advanced Modulation and Coding Schemes (MCS) to adapt to different network conditions.

The parameters required for this modulation are [28]:

- **Channel Bandwidth (CBW)**: Transmission bandwidth (CBW20, CBW40, CBW80, CBW160).
- **Modulation and Coding Scheme (MCS)**: Defines the modulation type and coding rate.
- **Channel Coding**: Error correction scheme (LDPC or BCC).

This table shows the modulation type and coding rate for each valid value of MCS (Table 2.1):

MCS	Modulation	Coding Rate	FEC
0	BPSK	1/2	LDPC/BCC
1	QPSK	1/2	LDPC/BCC
2	QPSK	3/4	LDPC/BCC
3	16-QAM	1/2	LDPC/BCC
4	16-QAM	3/4	LDPC/BCC
5	64-QAM	2/3	LDPC/BCC
6	64-QAM	3/4	LDPC/BCC
7	64-QAM	5/6	LDPC/BCC
8	256-QAM	3/4	LDPC/BCC
9	256-QAM	5/6	LDPC/BCC
10	1024-QAM	3/4	LDPC
11	1024-QAM	5/6	LDPC

Table 2.1. Modulation and Coding Scheme (MCS) in Wi-Fi HE-SU 802.11ax [28]

The waveform length remains fixed at 90,960 samples for every 32768 bits (4096 bytes), ensuring dataset consistency.

Wi-Fi VHT (802.11n/ac)

The Very High Throughput (VHT) mode of Wi-Fi 802.11n/ac [24] [29] is based on OFDM (Orthogonal Frequency Division Multiplexing), allowing higher data rates and improved

spectral efficiency. It supports wider bandwidths and advanced modulation schemes, making it a key technology in modern Wi-Fi networks.

The parameters required for this modulation are the same as for the previous modulation [30]:

- **Channel Bandwidth (CBW)**: Transmission bandwidth (CBW20, CBW40, CBW80, CBW160).
- **Modulation and Coding Scheme (MCS)**: Defines the modulation type and coding rate.
- **Channel Coding**: Error correction scheme (LDPC or BCC).

Only, the Modulation and Coding schemes are different. They are the ones shown in table 2.2, which presents the modulation type and coding rate for each valid value of MCS:

MCS	Modulation	Coding Rate	FEC
0	BPSK	1/2	LDPC/BCC
1	QPSK	1/2	LDPC/BCC
2	QPSK	3/4	LDPC/BCC
3	16-QAM	1/2	LDPC/BCC
4	16-QAM	3/4	LDPC/BCC
5	64-QAM	2/3	LDPC/BCC
6	64-QAM	3/4	LDPC/BCC
7	64-QAM	5/6	LDPC
8	256-QAM	3/4	LDPC
9	256-QAM	5/6	LDPC

Table 2.2. Modulation and Coding Scheme (MCS) in Wi-Fi VHT 802.11n/ac [30]

The waveform length is fixed at 93,120 samples, ensuring a standardized dataset structure for every (4096 bytes).

Wi-Fi Non-HT (802.11b)

The Non-High Throughput (Non-HT) mode of Wi-Fi 802.11b [23] is based on DSSS (Direct Sequence Spread Spectrum), enabling reliable low-data-rate transmission. It is widely used in legacy Wi-Fi networks and IoT applications due to its robustness in noisy environments.

Just like in the DSSS modulation, the only required parameter is **Data Rate**, which, defines the transmission speed (1, 2, 5.5, or 11 Mbps) and modulation, as explained in the DSSS subsection [31].

The waveform length for every 32760 bits (4095 bytes) depends on the data rate:

- For 1Mbps, the waveform length is 362472 samples.
- For 2Mbps, it is 182292 samples.
- For 5.5Mbps, the waveform length is 67632 samples.
- For 11Mbps, the length is 34872 samples.

2.3.2. Dataset Format and Organization (HDF5 Format)

For the organization of the waveforms generated, the HDF5 file format has been chosen. For each type of modification used, an HDF5 file will be generated to include the datasets, which are typed multidimensional arrays.

For our specific use case, we will actually have 2 different HDF5 files for each modulation: *bits_mod.h5* and *mod.h5*, where we will store the bits of the videos that will be modulated (or any other sequence of bits to be transmitted), as well as the waveforms generated from these bits, respectively for each type of modulation.

Since most of the waveforms are complex signals, they will be stored in the dataset using 2 channels: one for the real part of the waveform and another for the imaginary one.

Inside the *HDF5* files, all datasets must be stored in individual datasets with proper names. In our case, as each dataset is generated independently for one kind of modulation, only one dataset per file will be created, and all of them will be named *dataset* within each file. Additionally, each signal within the dataset will include an attribute specifying the *framesize* of the video used to generate that particular signal as an array of 2 values. In our case, this framesize will be consistent for all generated signals (independently from the modulation used), but we consider this is an important parameter to take into account, since it provides essential metadata linking the signal back to the original video frame structure, enabling consistent interpretation and processing of the waveform data, and could be useful in future works where our framework can be reused. In Figure 2.8 we can see one example of one of the generated datasets. In this case, a dataset of 16000x2x400 samples can be observed. This means that in this example dataset there are 400 signals of 16000x2 samples, corresponding to one complex signal of 16000 samples (first dimension corresponding to the *In-phase* component and the second to the *Quadrature* component). Also, we can see the corresponding bitstream for each of the generated signals, with 100.000 samples each. Additionally, in both, bits and waveform datasets, the *Framesize* attribute can be appreciated (50x50).

Also in the same folder as these files, there will be a *.mat* file along with a *json* file for each pair of HDF5 files, corresponding to the modulation and related parameters, which could be used as metadata for each of the signals in each HDF5 file (In Figure 2.9 an example of the structure of the resulting folder for one OFDM modulation can be seen). Each one of these files will have different parameters according to the corresponding

```
>> h5disp('OFDM_1.h5')
HDF5 OFDM_1.h5
Group '/'
    Dataset 'dataset'
        Size:  16000x2x400
        MaxSize:  16000x2x400
        Datatype:   H5T_IEEE_F32LE (single)
        ChunkSize:  []
        Filters:  none
        FillValue:  0.000000
        Attributes:
            'FrameSize':  50.000000 50.000000
>> h5disp('bits_OFDM_1.h5')
HDF5 bits_OFDM_1.h5
Group '/'
    Dataset 'dataset'
        Size:  100000x400
        MaxSize:  100000x400
        Datatype:   H5T_STD_I8LE (int8)
        ChunkSize:  []
        Filters:  none
        FillValue:  0
        Attributes:
            'FrameSize':  50.000000 50.000000
```

Fig. 2.8. Example of structure of generated HDF5 files

modulations, though some parameters will be common to all of them, such as:

- *mod*: word representing the modulation and main parameter used for each dataset, e.g. *OFDM_QPSK* (representing OFDM modulated with QPSK), *wifiVHT_2* (representing the wifiVHT standard with a MCS of 2).
- *Fs*: sampling frequency. It is important to note this because, although each modulation has one sampling frequency, oversampling might be used, so we need to know the Fs.
- *oversamplingFactor*: Oversampling factor used. In case oversampling was not used, this value will be 1 (default).
- *BW*: Bandwidth.
- *ChannelCoding*: Forward-error-correction (FEC) coding type specified as 'LDPC' for low-density parity-check (LDPC) coding or 'BCC' for binary convolutional coding (BCC).
- *payload*: The payload parameter represents the number of bits that can be modulated at each time. If the total number of bits to be encoded exceeds this value, the bits will be divided into fragments of *payload* bits, and each fragment will be modulated separately. The resulting waveforms will then be concatenated to form a single continuous waveform.
- *waveformLength*: number of samples of each of the concatenated waveforms. The resulting number of samples will be a multiple of this value. If oversampling was used, this number must be multiplied by the oversampling factor.

In figure 2.10 we can see an example of the *json* and *mat* files corresponding to the

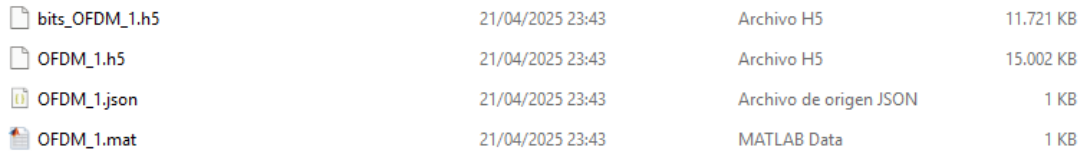


Fig. 2.9. Example of dataset folder structure

OFDM modulation used in the examples we are seeing.

Apart from these fields, these files will also have a specific field for the parameters of each modulation (explained before), as well as other parameters derived from these ones.

```
>> OFDM_1_params = load('OFDM_1.mat')

OFDM_1_params =

  struct with fields:

                  mod: 'OFDM_1024QAM'
                 type: 'OFDM'
           modulation: '1024QAM'
                   fs: 64000000
    oversamplingFactor: 1
                  cbw: 20000000
              payload: 53000
       waveformLength: 8000
    cyclicPrefixLength: 16
             FFTLength: 64
     guardBandCarriers: [2×1 double]
           numSymbols: 100
     subcarrierSpacing: 1000000
               DCnull: 0
           lengthBits: 300000
                  snr: 100
```

(a) *mat*


```
{"mod":"OFDM_1024QAM",
"type":"OFDM",
"modulation":"1024QAM"
,"fs":6.4E+7,
"oversamplingFactor":1,
"cbw":2.0E+7,
"payload":53000,
"waveformLength":8000,
"cyclicPrefixLength":16,
"FFTLength":64,
"guardBandCarriers":[6,5],
"numSymbols":100,
"subcarrierSpacing":1.0E+6,
"DCnull":false,
"lengthBits":300000,
"snr":100}
```


(b) *json*

Fig. 2.10. Example files for modulation parameters

2.4. Python Notebook for Signal Visualization from Dataset

The generated datasets, as explained before, consist of waveforms (signals) modulated according to different modulations or standards from several videos. Since all of the information contained is temporal signals, a tool for visualizing these signals should be very useful. For this purpose, a *Jupyter notebook* in Python has been developed.

Every kind of signal generated is different and has different parameters, all of them stored in *.mat* and *.json* files, easily readable in Python. These metadata files (the *.json* ones, specifically) are used in the notebook to get information about the signals and represent them.

Initially, in the notebook, all the available datasets in the selected folder are shown by running the `print_mods()` function (Figure 2.11). In case no folder path is passed to the function, it will assume that the folder is the one where the notebook is being run.

```
# In case current folder is not the datasets folder, add the path to the function: print_mods(path)
print_mods()
```

```
DSSS_1 :
  Modulation DSSS 1Mbps

DSSS_2 :
  Modulation DSSS 2Mbps

DSSS_3 :
  Modulation DSSS 5.5Mbps

DSSS_4 :
  Modulation DSSS 11Mbps

OFDM_1 :
  Modulation OFDM BPSK

OFDM_2 :
  Modulation OFDM QPSK

OFDM_3 :
  Modulation OFDM 16QAM

OFDM_4 :
  Modulation OFDM 64QAM

OFDM_5 :
  Modulation OFDM 256QAM

OFDM_6 :
  Modulation OFDM 1024QAM
```

Fig. 2.11. List available datasets in folder

Then, a dropdown is added in case the user wants to get more information about one of the modulations specifically (Figure 2.12).

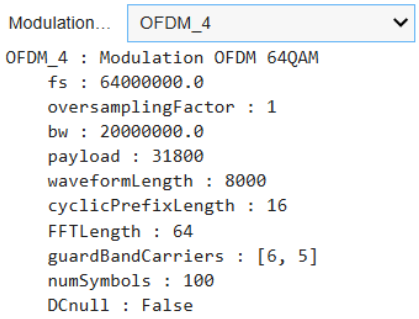
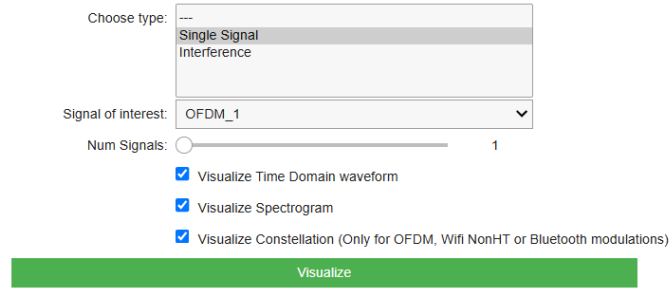


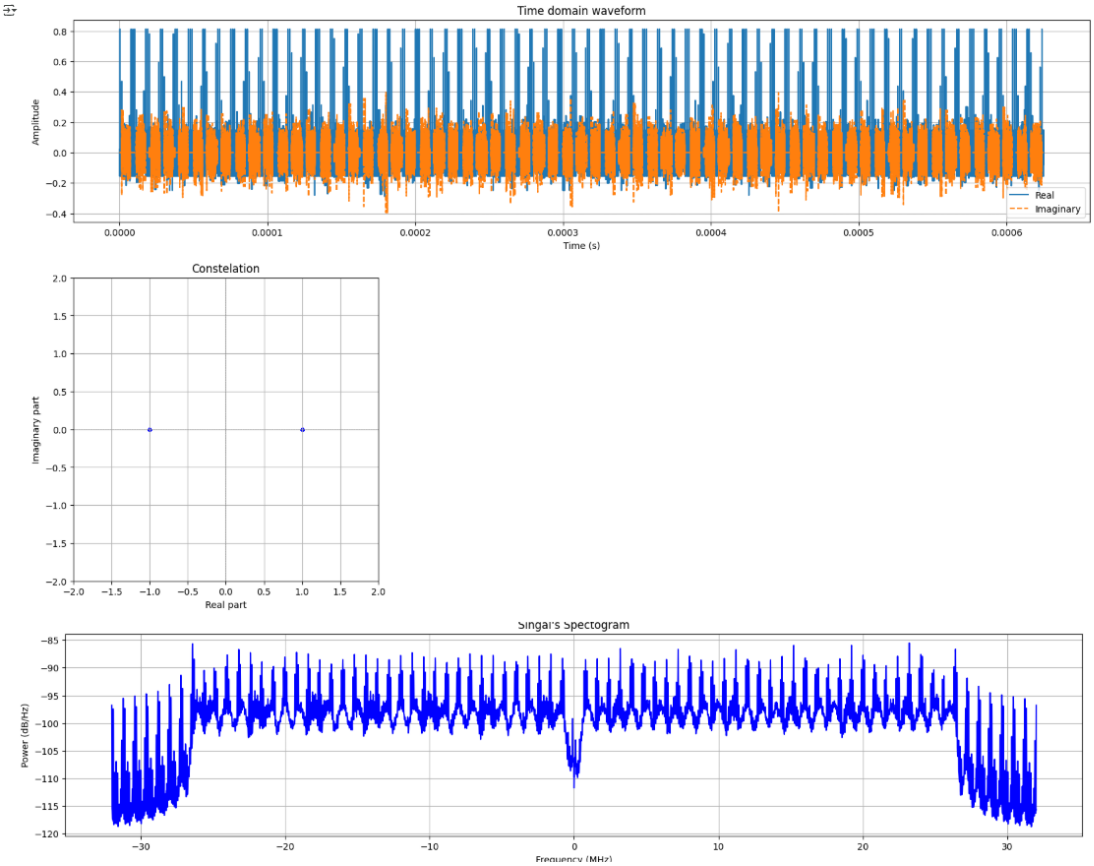
Fig. 2.12. Display metadata about a selected modulation

Finally, there is one last cell where we can eventually visualize the signals in the time domain as well as their spectogram (along with the constellation diagram in case it is available, as not all modulations or standards are based on constellation symbols). After running this last cell, a first dropdown is displayed where the user must choose between *Single Signal* or *Interference*. By choosing the first one, another dropdown will appear, asking for which kind of modulation the user wants to display. Once selected, some other options can be chosen, just like the number of signals the user wants to display (from the kind of modulation already selected) as well as the graphics the user wants to display. The signals shown will be the n first signals of the dataset. In Figure 2.13a, the interface of this application is shown. In Figure 2.13b, an example of the plotted graphics can be

seen.



(a) Options for visualizing the signals

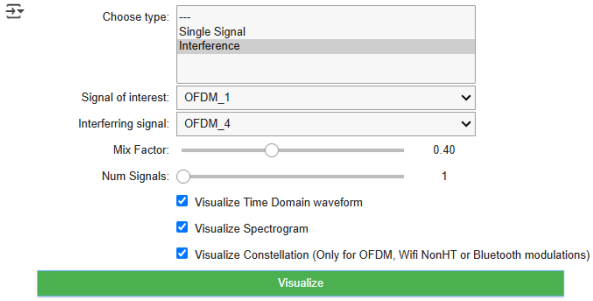


(b) Example of signal from dataset plotted

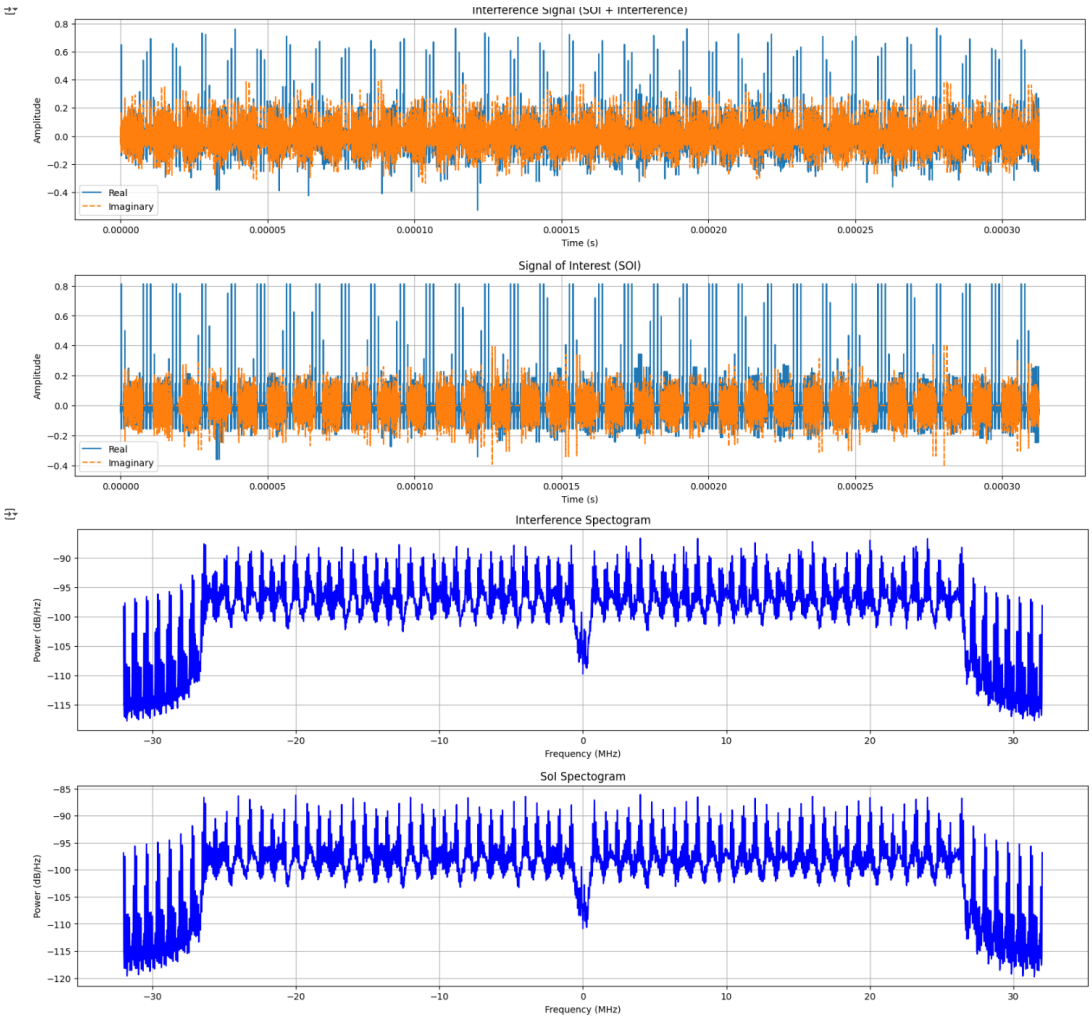
Fig. 2.13. Cell for selecting and visualizing the signals

If the *Interference* option is selected, the process is similar, although in this case the user will have to choose 2 types of signals to interfere with each other and also select the *mix_factor*, a factor that defines the proportion of power the interfering signal will have with respect to the Signal of Interest (main one).

The *Interference* option is important because, as discussed in the dataset generation section (Section 2.3), our dataset does not inherently include interference signals. However, as highlighted in the various existing datasets from the State of the Art (Section 1.1), interference is a key application for RF datasets. By incorporating this feature in the final notebook, users can visualize the different types of interference that can be generated with



(a) Options for visualizing the interference of two signals from the dataset



(b) Example of interference signal and SoI from dataset plotted

Fig. 2.14. Cell for selecting and visualizing the interference signals

the dataset. After reviewing the results, they can proceed to handle these interferences using the Python functions detailed in the following section.

2.5. Generation of Interferences

The process of generating interference signals is crucial for augmenting the original dataset with additional complexities that simulate real-world scenarios. By combining clean signals (referred to as Signal of Interest, or SoI) with interference signals, new samples are created with controlled interference levels. These augmented samples can then be used to train models aimed at mitigating such interferences.

Firstly, the script requires the user to specify three directories: one containing the SoI signals, another containing the signals to be used for interference and a final one for storing the generated interferences. These directories should contain the respective datasets in .h5 file format. As for the metadata or bits dataset files, they should only be present in the SoI folder, as the metadata for the interfering signals is not relevant for this process.

The attenuation factor(s) to be applied to the interference signals must be specified at the beginning of the script. This can be provided as a single floating-point value or as an array of floating-point values. The function will then generate all possible combinations of interferences, which will be the product of the number of signals in the Signal of Interest (SoI) folder, the number of signals in the interfering folder, and the specified attenuation factors.

The interference signals are typically modeled to represent common types of noise or interference found in wireless communication systems. Although this specific function is designed to work with datasets generated by the previously described application, the interference signals can be sourced from any similar *.h5* dataset. The goal is to simulate realistic interference scenarios that challenge the performance of signal processing algorithms by introducing noise or unwanted signals into the original clean signals.

2.5.1. Process of Generating Interference Signals

Interference is added to the SoI signals using the Python function `create_interference_dataset.py` from the `interf_utils.py` module. This function works as follows:

- **Signal Length Adjustment:** First, the function ensures that both the SoI and interfering signals have the same length. If their lengths do not match, the interfering signal is adjusted using the `adjust_signal_length` function, either by trimming or repeating the signal to match the length of the clean signal.
- **Interference Application:** Once the lengths are aligned, the interference signal is applied to the clean signal with a specified attenuation factor. This attenuation factor controls the strength of the interference relative to the clean signal.

- **Saving the Resulting Signal:** The resulting signal, which is the SoI combined with the interference, is saved into a new `.h5` file. This file is stored in the specified directory with a name derived from the original clean signal file name.

2.5.2. Storage of Generated Interference

The generated interference signals are stored in a new directory structure under the base output directory, which is specified in the `generate_interferences.py` script. For each pair of clean and interference signals, the following steps are taken:

- **Subdirectory Creation:** A subdirectory is created for each interference level (attenuation factor) and each pair of clean and interference signal files. The subdirectory is named using the format `interference_SoI_file_att`, where `SoI_file` is the base name of the SoI file, and `att` is the attenuation factor (e.g., `interference_signal1_050` for 50% attenuation).
- **File Saving:** The resulting interfered signals are saved in the corresponding subdirectory with the SoI's name. These files are saved as `.h5` files, preserving the same format as the original SoIs.
- **Auxiliary File Copying:** In addition to the interference signals, auxiliary files related to the clean signals (such as `.json`, `.mat`, and `bits_*.h5` files) are copied into the subdirectory. These files contain metadata and the original bits dataset, ensuring that all necessary data is preserved and accessible.

2.5.3. Final Folder Structure

The final structure of the output directory will look like the following:

```
interferences_folder/
        interference1_050/
            SoI.h5
            SoI.json
            SoI.mat
            bits_SoI.h5
        interference1_075/
            SoI.h5
            SoI.json
            SoI.mat
            bits_SoI.h5
        interference2_050/
            SoI.h5
            SoI.json
            SoI.mat
```

```
    bits_SoI.h5
interference2_075/
    SoI.h5
    SoI.json
    SoI.mat
    bits_SoI.h5
...
```

Here, each subdirectory corresponds to an interfering signal at a specific attenuation factor and its datasets (despite being named after the SoI) correspond to the resulting interference dataset. The interference dataset, along with the related files from the SoI, are stored together for easy access.

In Figure 2.15 an example of the resulting folder structure is shown. The *SoI* and *interfering* folders correspond to the initial folders with the *clean* signals to be mixed, while the *interference* folder is the output folder where the interferences will be stored. In this example, we see we have 12 files in the *SoI* folder, corresponding to 3 signals (remember that for each signal there is 4 files associated: *.mat*. *.json* and 2 *.h5*, one for the waveforms and one for the original bits). We want to mix these signals with 5 other signals (20 files in *interfering* folder), which are 4 different configurations of OFDM and 1 of Bluetooth. And finally, 2 mixing factors: 0.5 and 0.75. This way, the *interference* folder has 10 subfolders (2 mixing factors x 5 interfering signals) with 4 datasets each (12 files, corresponding to the SoIs).

Usage of the Generated Interference Dataset

Once the interference signals are generated, they can be used in training and testing machine learning models, particularly those designed to mitigate interference in communication systems. By including these generated interference signals in the dataset, the models can be exposed to more realistic, noisy conditions, improving their robustness and performance in real-world scenarios.

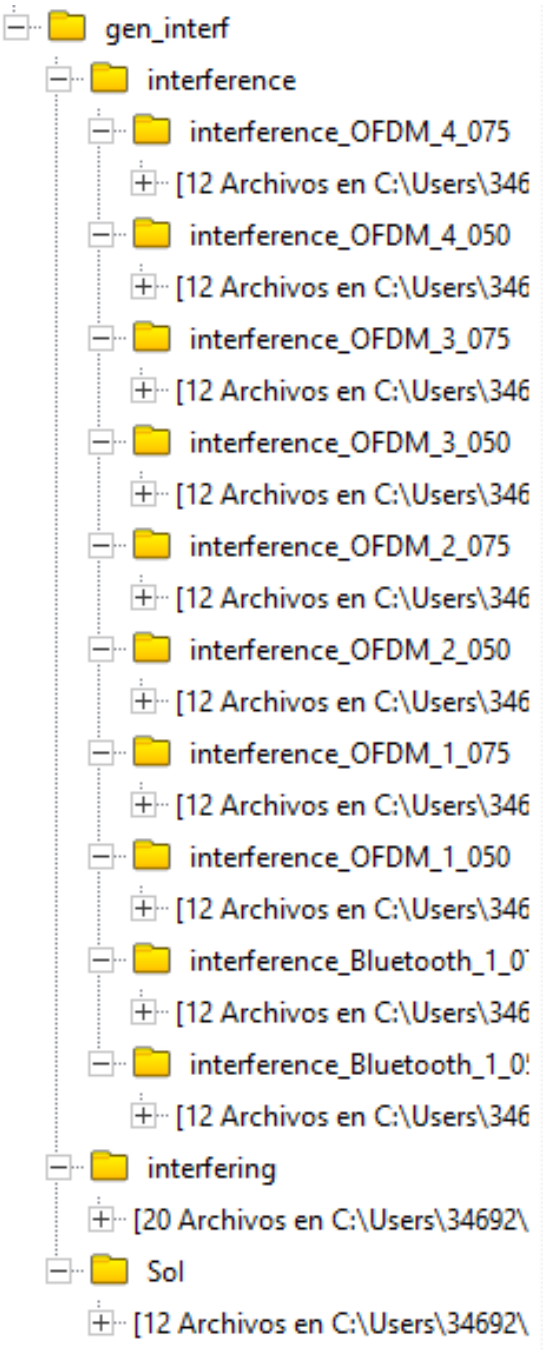


Fig. 2.15. Folder Structure for Interference Generation

3. Use Case: Machine Learning Evaluation for Interference Mitigation

In this chapter, we present a practical use case to demonstrate the applicability and utility of the proposed dataset for machine learning-based interference mitigation in wireless communication systems. Rather than focusing solely on performance benchmarks, the objective is to validate the dataset's capacity to support the development and evaluation of deep learning models within the scope of RF signal processing.

3.1. Machine Learning Model: UNet

As a first approach to evaluate the applicability of the generated dataset, a UNet-based architecture was chosen due to its proven success in similar tasks. UNet has demonstrated robust performance in various signal separation and interference rejection scenarios, particularly in single-channel settings, where the signal of interest and interference overlap in the same time-frequency domain and are received through a single input channel, typically from a single antenna. Its architecture has been widely validated in the context of RF interference rejection tasks, where it has been shown to effectively separate signals and mitigate interference. Given its demonstrated success with other datasets, such as the one presented in [18], UNet was considered an ideal choice for evaluating the performance of the generated dataset in interference mitigation tasks.

Originally proposed for biomedical image segmentation, the UNet model architecture is particularly well-suited for structured prediction tasks that require detailed spatial or temporal localization. In the context of RF signal processing, the temporal dimension of the signal plays a key role, and convolutional layers are adapted to operate over one-dimensional sequences.

Figure 3.1 illustrates the UNet architecture adopted for this study. The model employs 1D convolutional layers to extract temporal features from baseband time-series data. Since the input signals are complex-valued, the real and imaginary components are handled as separate input channels, allowing the network to learn complex interactions without requiring complex-valued operations.

The architecture consists of an encoder-decoder structure with skip connections. The encoder path progressively reduces the temporal resolution via convolutional and downsampling layers, capturing increasingly abstract representations. The decoder path then reconstructs the signal using upsampling layers, while reintroducing fine-grained details through skip connections that concatenate encoder features at matching resolutions. This structure allows the network to integrate both local and global temporal information effectively, which is crucial in recovering signals degraded by interference or noise.

The base UNet code and functions were originally sourced from [1], where the model was implemented using TensorFlow. However, in this study, we sought to adapt the model

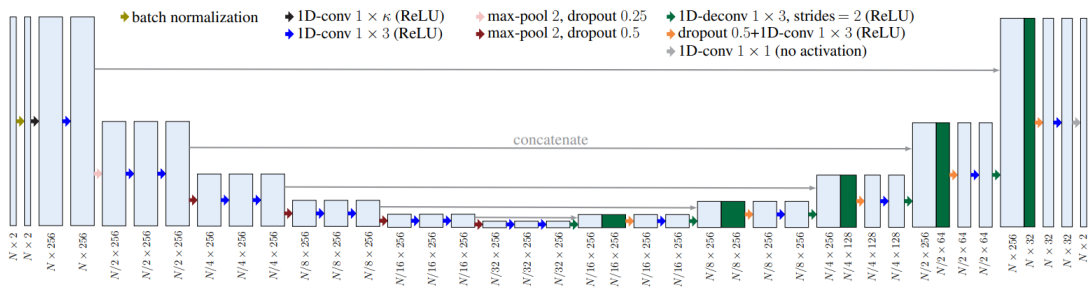


Fig. 3.1. Architecture of the UNet DNN used in our Use Case [18].

to our dataset and specific application needs, as well as to transition the implementation to PyTorch. The reason for this transition was primarily driven by PyTorch's more intuitive and flexible interface, which is particularly advantageous for research and development. PyTorch's dynamic computational graph also allows for easier debugging and more control over the model, making it a more suitable option for experimentation. The modified scripts used in this work, including those for training, inference, and dataset preparation, are available in the already mentioned GitHub repository [32], ensuring transparency and reproducibility of the results.

It is interesting to discuss the choice of the kernel size $\kappa = 101$ for the first convolutional, which is not arbitrary. According to the findings in [18], κ must be carefully chosen to align with the effective correlation length of the signals involved. This insight comes from the recognition that RF signals exhibit temporal structures, both in the signal of interest (SOI) and the interference, which extend over multiple samples.

If the kernel size is too small, the network may fail to capture the necessary temporal dependencies required for effective separation. Conversely, a kernel size that covers at least the temporal span of the autocorrelation of the SOI and the cross-correlation between the SOI and the interference ensures that the network can access and leverage these structures. As a result, adjusting κ to match or exceed this correlation span significantly improves performance, in some cases achieving near-optimal MMSE performance in structured scenarios, such as cyclostationary Gaussian signals.

For this study, we chose a comparatively large kernel size for the first convolutional layer (e.g., $\kappa = 101$), as this enables the network to capture long-term temporal dependencies crucial for recovering signals affected by overlapping interference and noise. This choice was empirically validated and aligns with domain-informed architectural modifications proposed in the original UNet adaptation for RF signal separation.

The model's objective is to learn a mapping from corrupted input signals (e.g., WiFi degraded by Bluetooth interference) to their clean counterparts. This is conceptually similar to source separation, where the goal is to isolate the desired signal from unwanted overlapping transmissions.

This use case simulates a realistic receiver scenario, in which some signals coexist in time and frequency with other transmissions in the same band (ISM in this case). The

UNet is trained to reconstruct the SoI (signal of interest) signal from these interfered signals. The model's effectiveness is later evaluated through signal reconstruction quality and communication-level metrics, such as Bit Error Rate (BER) and Mean Squared Error (MSE).

This architecture serves as a proof-of-concept to validate the generated dataset's utility, establishing a baseline for future experiments with more advanced neural architectures or multi-signal input scenarios.

3.2. MATLAB Application for Signal Demodulation and Visualization

In order to facilitate the analysis and validation of the wireless signal datasets in a real ML application, a dedicated MATLAB tool has been developed. This application provides two distinct modes of operation, each tailored to different user needs and levels of analysis complexity.

The first mode, referred to as the *Simple Visualization*, allows users to demodulate a selected signal and directly compare the demodulated bits with the original transmitted bits. The signal to be demodulated can be either a *clean* signal or an interferenced one, as well as any other the user may want. The only thing is that the metadata files (*.json* or *.mat*) must be in the same folder, along with the dataset with the original bits, following the folder and file strcture that has been followed throughout this project. This mode is ideal for quick validation and basic inspection of the signals and their integrity.

The second mode, called *Inference Visualization*, is designed for more advanced scenarios where signals obtained through inference methods (such as interference mitigation, signal separation, or other enhancement algorithms) are compared side-by-side with original signals and their respective interference. This mode enables a comprehensive comparative analysis involving three synchronized visualizations: the inferred signal, the interference signal, and the original signal from its bits, providing deeper insights into the effectiveness of the applied processing techniques.

The initial screen of the application allows the user to select between these two modes, streamlining the workflow depending on the analysis requirements (Figure 3.2).

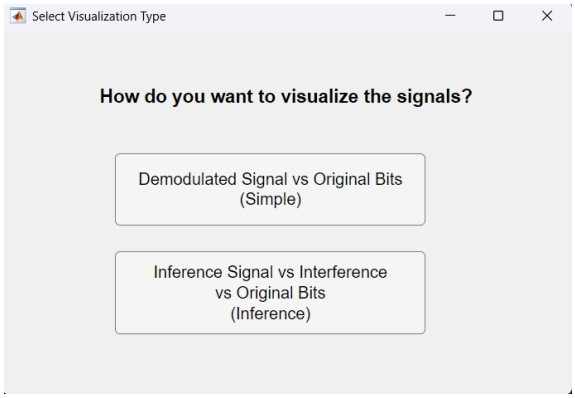


Fig. 3.2. First screen of the MATLAB demodulation and visualization application.

In the following subsections, we will first describe the Simple Visualization mode in detail, covering the user interface, signal selection, demodulation process, and visualization features.

It is important to point out that the application currently supports demodulation for a selected set of modulation schemes, including PSK, QAM, OFDM, and Bluetooth signals. These choices are based on the availability of well-established demodulation algorithms and the feasibility of implementation within the scope of this project. More complex standards such as Wi-Fi and 5G, which involve intricate protocols and signal processing chains, have been deliberately excluded to maintain focus on core modulation techniques and to ensure robustness and reliability of the demodulation tool. The implemented demodulation functions are developed by adapting and inverting the corresponding modulation routines provided by the MATLAB Wireless Waveform Generator [19] toolbox, providing a consistent and accurate approach across supported schemes.

3.2.1. Simple Visualization Mode

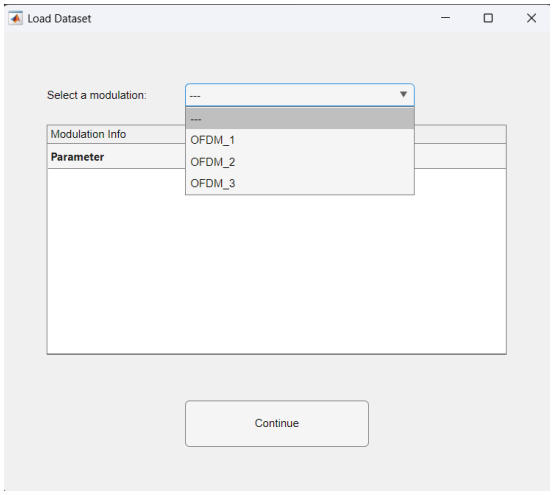
Upon selecting the Simple Visualization mode, the user is first prompted to choose the dataset folder they wish to analyze. As previously described, this folder must adhere to the established dataset structure, containing the modulated signal files (*.h5*), the corresponding original bits datasets, as well as the associated `.mat` and `.json` metadata files for each modulation or standard.

After selecting the appropriate folder, the application proceeds to the *Dataset Selection* screen. Typically, each dataset folder contains multiple modulation types, each corresponding to a different dataset file. This screen automatically scans the selected folder and retrieves all available modulation datasets. The user is then presented with a dropdown menu listing the detected modulation schemes (Figure 3.3a). Upon selecting a modulation from the list, the relevant modulation parameters are displayed, allowing the user to review key characteristics of the chosen dataset. This preview functionality helps ensure that the user selects the desired modulation configuration before continuing (Figure 3.3b).

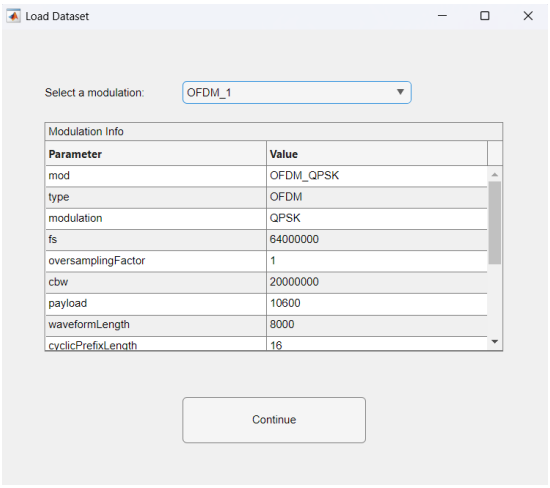
Only after confirming the modulation choice can the user proceed to the next stage, which is to select the specific signals to visualize (and demodulate) within the chosen dataset.

This next window shows the total number of signals available and lets the user specify how many signals to process. Signals can be chosen manually by entering their indices or selected randomly via a checkbox, with the interface adjusting accordingly to show or hide manual selection fields. This interface is shown in Figure 3.4.

To ensure valid input, duplicate signal selections are checked and flagged to the user. The window also provides standard navigation buttons to proceed to the next step or return to the previous screen to select any other of the modulations without needing to exit the tool.



(a) List of available modulations in folder



(b) Information about selected modulation (OFDM_1)

Fig. 3.3. Window for Dataset Selection

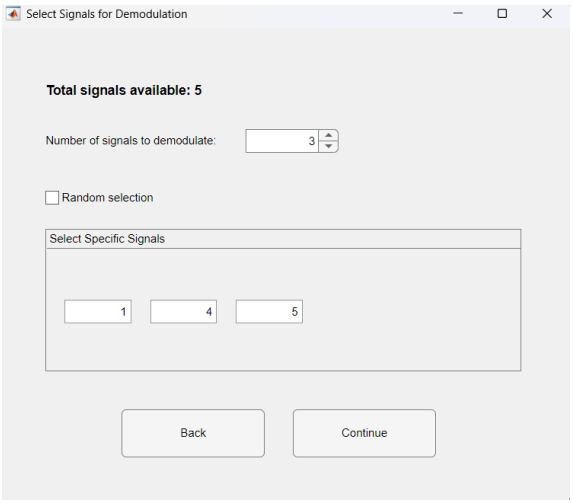


Fig. 3.4. Interface for selecting signals to visualize

When clicking on the *Next* button the last window pops as the primary interface for demodulating and visualizing the user-selected signals. Upon loading the modulated signals dataset (*.h5* file), the corresponding original bitstreams (*bits_*.h5*), and modulation parameters (*.mat* file), the application begins demodulating the signals.

This interface displays the current signal index and calculates the Bit Error Rate (BER) between the original and demodulated bits, providing a quantitative measure of demodulation accuracy. Additionally, it offers synchronized playback of videos reconstructed from both original and demodulated bitstreams, enabling direct visual comparison (Figure 3.5).

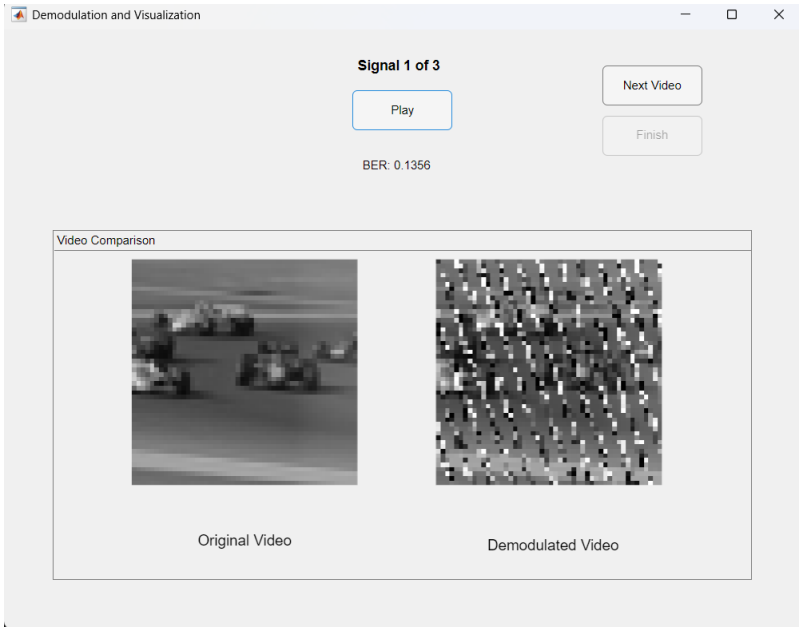


Fig. 3.5. Original video vs Demodulated Signal Interface

In Figure 3.5, an example visualization is shown of an OFDM signal modulated with QPSK, combined with a Bluetooth interference signal attenuated by a factor of 0.5. As illustrated, the resulting Bit Error Rate (BER) is 0.1356, which introduces some noise to the reconstructed video while still allowing the original image to be discernible underneath.

The primary objective of this application is to add an extra layer of measurement beyond traditional numerical metrics (such as loss functions used in machine learning) by providing a **visual and intuitive assessment** of the signal processing performance. This visual layer enables users to better understand the impact of interference and algorithm effectiveness in a more tangible and interpretable way.

3.2.2. Inference Visualization Mode

In contrast to the Simple Visualization mode, the *Inference Visualization* mode requires the user to select two dataset folders: first, the folder containing the signals with interference (referred to as the interference signals), and second, the folder containing the

processed signals, hereafter called the *inference signals*. The original clean signal is not selected explicitly as a separate folder; instead, it is obtained from the dataset of original bits associated with either of the two selected folders, since these should be identical.

Although this mode is described in the context of interference and inferred signals, it is flexible enough to handle other types of signal processing or video modifications, as long as the datasets share the same structure and naming conventions. This allows for comparing different versions of the same signals or videos processed by various algorithms.

Prior to folder selection, the user is presented with an instructional screen indicating the required order for folder selection—first the SoI folder and then the inference folder (see Figure 3.6). This ensures clarity and reduces user error.

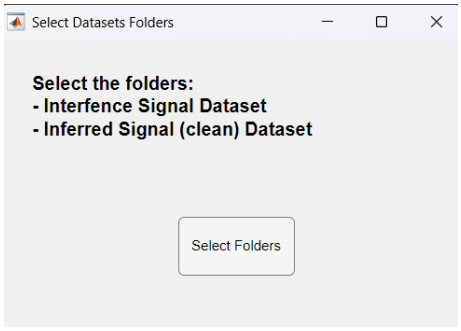


Fig. 3.6. First screen prior to selecting folders.

It is essential that both selected folders contain datasets with the same file structure and, more importantly, that they share identical file names. This compatibility guarantees that the application can match corresponding signals between the two datasets correctly. By enforcing this naming convention, when the user selects modulations in the subsequent screen (see Figure 3.3), the available modulation types will be consistent across both datasets, simplifying the comparison process.

Similarly, in the signal selection window (see Figure 3.4), the user is restricted to choosing signals that exist in both datasets, ensuring that the selected indices correspond exactly to matching signals. To guarantee this synchronization, it has already been verified that all files in both folders have the same names, and then the original bits datasets from both folders are compared to confirm that the original signals themselves are identical. This process ensures that the signals selected for demodulation and comparison correspond precisely across both datasets, enabling accurate and meaningful visualization.

The final and main visualization screen changes significantly from the simple mode. Here, three synchronized video visualizations are presented side by side: the original clean signal, the interference-affected signal, and the inference (processed) signal. This interface allows the user to directly compare the effects of processing algorithms on the signal quality and the resulting video reconstruction (Figure 3.7).

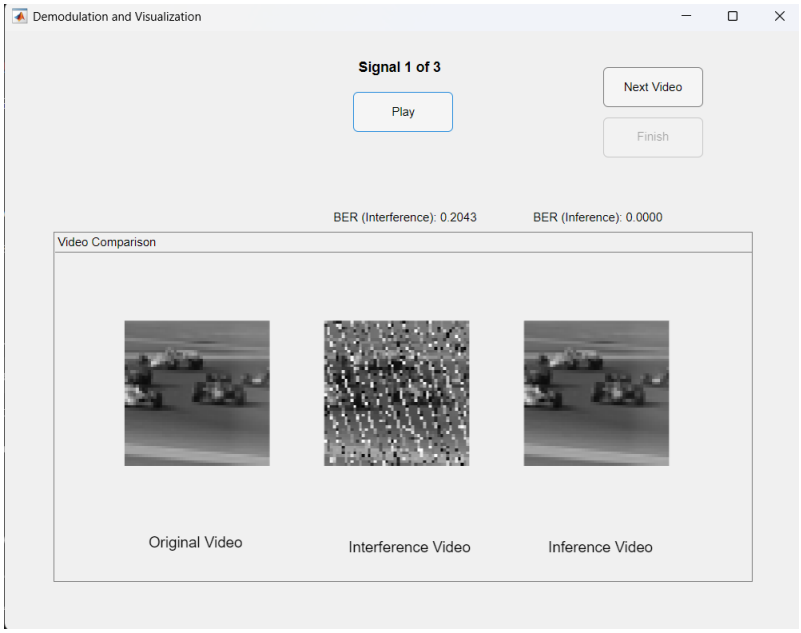


Fig. 3.7. Original video vs Demodulated Interference Signal vs Demodulated Inferred Signal Interface

In Figure 3.7, an example for this interface is presented, this time with a different modulation-interference combination. The Signal of Interest (SoI) is an OFDM signal modulated with 64QAM, and the interference signal is an OFDM signal modulated with 256QAM. Similarly to thr previous example, the user is presented with three synchronized video streams: the original signal, the interference signal, and the inference signal after processing. The interference signal shows a Bit Error Rate (BER) of 0.2043, introducing noticeable noise to the reconstructed video, but still retaining some visual representation of the original image. After applying the corresponding algorithm (in this case an interference mitigation technique), the inference signal yields a completely clean image, achieving a BER of 0. This demonstrates the success of the applied processing algorithm not only through traditional model performance metrics but also through the comparison of BER values. Furthermore, this approach allows the user to visually assess the improvement in signal quality, thereby fulfilling the objective of this tool: offering both numeric and visual feedback on the effectiveness of signal processing methods.

3.3. Evaluation of UNet Model over Generated Dataset for Interference Mitigation

3.3.1. Dataset Description and Interference Scenarios

To evaluate the proposed dataset under controlled but realistic conditions, a simplified use case was defined where only Orthogonal Frequency Division Multiplexing (OFDM) signals are used as the signal of interest (SoI). This decision was motivated by the balance OFDM strikes between structural complexity and tractability. While simpler modulation

schemes such as PSK or QAM offer limited variation, fully standardized protocols like WiFi or Bluetooth introduce considerable additional complexity due to protocol overhead and signaling structures. Thus, OFDM provides a versatile yet manageable middle ground for preliminary evaluations.

The dataset includes three distinct types of OFDM signals, each employing a different modulation scheme: QPSK, 64-QAM, and 1024-QAM. These three types were selected to cover a wide range of constellation complexities, enabling the model to generalize to various signal structures using only a limited set of examples. Each waveform was generated with consistent OFDM parameters such as a cyclic prefix length of 16, an FFT length of 64, a subcarrier spacing of 1MHz, and a channel bandwidth of 20MHz. All signals were sampled at 64MHz with no oversampling.

The parameters for each signal type are summarized below:

- **OFDM with QPSK:** payload of 10,600 bits per waveform segment, resulting in a total of 10 concatenated segments to reach 100,000 bits. The final waveform length is therefore $10 \times 8000 = 80,000$ samples.
- **OFDM with 64-QAM:** payload of 31,800 bits per waveform segment, leading to 4 concatenated segments. The total waveform length is $4 \times 8000 = 32,000$ samples.
- **OFDM with 1024-QAM:** payload of 53,000 bits per segment, requiring 2 concatenated segments, yielding a total of $2 \times 8000 = 16,000$ samples.

To construct the dataset, 40 video samples were segmented into 10 sub-videos each, yielding a total of 400 training instances for each modulation configuration. Each instance contains one of the OFDM signals described above, embedded in a video-derived structure, with metadata such as modulation type, waveform parameters, and bit length stored in accompanying JSON or MATLAB '.mat' files.

Interference Dataset

To assess the robustness of the UNet model and the generalization capabilities of the dataset, an interference dataset was constructed using a diverse set of signals not included in the original training set. This allowed for the emulation of several realistic interference scenarios with varying degrees of similarity to the signals of interest (SoI).

For generating interfering signals, two groups of video data were used:

- 10 videos selected from the same pool as the training dataset (i.e., reused video sources).
- 25 videos from a disjoint set, ensuring non-overlapping content with the SoI signals.

Each of these 35 videos was divided into 10 subvideos (each containing 5 frames), resulting in 350 distinct interference instances. These were then paired with the first

350 SoI signals (400 SoI signals originally, but 350 were used to match dimensions) to generate interfered signal pairs used in evaluation.

The interference signals were designed to represent three distinct categories:

1. OFDM signals with same modulation as one set of the SoI signals: **1024-QAM.**
2. OFDM signals with different modulation types: **256-QAM.**
3. Non-OFDM signals: **Bluetooth BR (Basic Rate).**

All OFDM-based interference signals were created using the same waveform generation process as for the SoI, but with one of two parameter configurations:

- The original OFDM parameters used in training.
- A modified set of parameters, differing in aspects such as FFT length, cyclic prefix length, etc. Also, some noise was introduced when generating, selecting a SNR of 10dB instead of the original 50dB (or 100dB in case of OFDM 1024QAM) (noiseless) we were using.

Table 3.1 outlines the comparison between the initial and modified OFDM parameter configurations:

Parameter	**Original Configuration**	**Modified Configuration**
FFT Length	64	16
Cyclic Prefix Length	16	8
Subcarrier Spacing	1MHz	1MHz
Number of Symbols	100	40
SNR	50dB/100dB	10dB

Table 3.1. Comparison of OFDM Parameter Configurations for Interference Signals

In addition to OFDM signals, Bluetooth BR (Basic Rate) signals were introduced as a fundamentally different modulation type, representative of frequency-hopping spread spectrum (FHSS) systems that coexist with WiFi (which is based mainly on OFDM) in the ISM band.

To simulate varying interference power levels, each interfering signal was scaled using one of two attenuation factors: 0.5 or 0.75. This models different signal-to-interference scenarios and introduces diversity in interference intensity.

This setup allows for evaluating three key generalization dimensions:

- **Intra-class interference:** OFDM vs. OFDM (same modulation, different parameters).
- **Inter-class OFDM interference:** OFDM vs. OFDM (different modulation types).

- **Cross-protocol interference:** OFDM vs. Bluetooth BR.

These scenarios are critical for testing whether the model can distinguish and recover the original signal when exposed to modulations and signal characteristics not seen during training.

3.3.2. Evaluation Metrics

To quantitatively evaluate the performance of the UNet model in reconstructing signals from interfered observations, two primary metrics were employed: the Mean Squared Error (MSE) and the Bit Error Rate (BER). These metrics capture different but complementary aspects of model performance: numerical similarity in signal space, and correctness in digital communication terms, respectively.

Mean Squared Error (MSE)

The Mean Squared Error (MSE) [33] is a widely used loss function in regression tasks and neural network training. It measures the average of the squares of the differences between predicted values and the true values. Given a model output $\hat{x}$ and the ground-truth signal x, both of length N, the MSE is computed as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3.1}$$

In the context of signal recovery, x represents the original clean signal, while $\hat{x}$ is the signal estimated by the neural network. A lower MSE indicates that the reconstructed signal is closer to the original in a Euclidean sense, which generally reflects better performance.

While MSE is sensitive to amplitude and phase mismatches, it does not directly reflect the impact of reconstruction quality on communication performance. Therefore, MSE is often complemented by domain-specific metrics in communication systems.

Bit Error Rate (BER)

The Bit Error Rate (BER) [20] is a fundamental metric in digital communications, used to measure the proportion of received bits that are incorrectly decoded. Given a transmitted bit sequence $b = [b_1, b_2, ..., b_M]$ and a received bit sequence $\hat{b} = [\hat{b}_1, \hat{b}_2, ..., \hat{b}_M]$, the BER is defined as:

$$\text{BER} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{1}_{\{b_i \neq \hat{b}_i\}} \tag{3.2}$$

where $\mathbb{1}_{\{b_i \neq \hat{b}_i\}}$ is an indicator function that returns 1 if the bit b_i is incorrectly received, and 0 otherwise.

In RF signal recovery tasks, BER serves as a highly interpretable and objective metric. It not only quantifies how well the model reconstructs the waveform, but also evaluates whether the demodulated data is correct — a core requirement in communication systems. Importantly, BER evaluation is performed after demodulating the recovered waveform, thereby assessing the end-to-end impact of interference and recovery on communication integrity.

Moreover, using BER in this context offers a unique advantage: it enables a direct link between quantitative analysis and qualitative visual verification. Since each waveform corresponds to a video segment, visual inspection of the recovered video (e.g., blank vs. intelligible frames) can be used to cross-check and interpret BER results. This dual evaluation approach enhances the reliability and interpretability of the experimental conclusions.

3.3.3. Training and Testing Setup

The proposed model is trained using a two-stage procedure, structured to first extract the inherent characteristics of the signal of interest (SoI) and later focus on interference mitigation. The input and output of the model are both complex-valued time-domain signals, represented as two real-valued channels corresponding to the in-phase (I) and quadrature (Q) components, hence the model is designed to process two input and output channels.

Stage 1: Clean Signal Autoencoding

In the first training stage, we aim to train the UNet model as an autoencoder. This means the model is provided with clean OFDM signals as both the input and the expected output. Specifically, we train with 400 clean signals for each of three OFDM modulation types: QPSK, 64-QAM, and 1024-QAM, totaling 1200 training samples. These modulation schemes were selected as representative samples covering a wide range of constellation complexities. In practical wireless communication systems like WiFi, OFDM is widely used with modulations from BPSK to 1024-QAM, so by selecting a subset from the low, medium, and high complexity range, we aim to induce generalization over all possible intermediate modulations.

As indicated, the model is firstly trained on OFDM signals with QPSK modulation, followed by 64-QAM, and finally 1024-QAM, making the model to be progressively exposed to more intricate modulation schemes. This strategy ensures that the model is well-equipped to generalize across a broad spectrum of signal types, providing a robust foundation for effectively addressing interference in subsequent stages.

This setup encourages the model to learn a compressed latent representation of OFDM signal structures, exploiting the UNet's encoding path to capture spectral and temporal dependencies, and then reconstructing the original waveform through the decoding path. The learning objective is to minimize the Mean Squared Error (MSE) between input and output, helping the model abstract the typical temporal and spectral characteristics of OFDM signals.

Data is loaded from HDF5 files using a custom PyTorch dataset class. Each signal is converted to a `torch.FloatTensor` of shape $(2, N)$ where N is the number of samples, and the batch size is set to 16. We use the Adam optimizer with a learning rate of $3 \cdot 10^{-4}$, and employ gradient scaling via PyTorch's `GradScaler` for mixed-precision training on CUDA devices. Early stopping is implemented based on validation loss, with a patience of 10 epochs to prevent overfitting. The dataset is split into 90% training and 10% validation.

After this training, the model is evaluated on unseen signals, including different modulations such as 16-QAM or BPSK, to verify whether it has acquired transferable representations of general OFDM behavior.

Stage 2: Interference Mitigation

In the second stage, the model transitions from the autoencoder task in Stage 1 to interference mitigation, where the goal is to suppress interference and recover the clean Signal of Interest (SoI). The model used in this stage is initialized with the weights from the final model trained in Stage 1, ensuring that it retains the learned representations of the OFDM signal structure. This prior knowledge from Stage 1 provides the foundation for the model to build upon as it adapts to more complex tasks involving interference. In this stage, the model is trained on signals that are corrupted by various interference types, while the clean SoI remains the target output. By leveraging the pre-learned signal structure, the model is better equipped to handle interference scenarios and recover the clean signal even under different distortion types.

The interference dataset includes signals with various OFDM modulations (e.g., 256-QAM and 1024-QAM with alternate parameters), as well as classical Bluetooth BR signals. Two attenuation levels are considered (0.5 and 0.75) to simulate varying interference strengths. The clean and interference datasets are matched using their JSON metadata files, ensuring that each pair corresponds to the same signal except for the added distortion.

A custom dataset class `HDF5DenoisingDataset` is used to load matching pairs of clean and interfered signals. These are converted into batches with the same configuration as before, and training proceeds similarly with early stopping and MSE as the loss criterion. The dataset is split into training (90%), validation (5%), and test (5%) sets.

During inference, the model processes interfered signals to generate restored versions, which are stored in HDF5 format for later quantitative and qualitative evaluation. This

step validates whether the model has effectively learned to separate and reconstruct the SoI even under different interference types and configurations.

3.3.4. Results and Analysis

This subsection presents the outcomes obtained from the two-stage training process described previously, alongside a detailed interpretation of the results. First, we evaluate the model's performance in reconstructing clean OFDM signals from the initial autoencoder training, assessing its ability to generalize across different modulation schemes. Then, we analyze its effectiveness in mitigating interference through the denoising training phase, where the model aims to recover the original signal from degraded inputs. The evaluation is based on both objective quantitative metrics (MSE and BER) and qualitative visual inspection of the regenerated signals, allowing for a comprehensive assessment of the model's behavior under various signal conditions.

Stage 1: Clean Signal Autoencoding

Before evaluating the model's capacity to generalize or mitigate interference, it is essential to assess its training behavior. This first analysis focuses on the evolution of the training and validation loss during Stage 1, where the UNet model is trained as an autoencoder on clean OFDM signals. The training was carried out in three sequential phases, each corresponding to a specific OFDM modulation scheme: QPSK, 64QAM, and 1024QAM. This setup allows the model to gradually learn signal structures of increasing complexity. The performance during these three phases is visualized in the following figures: Figure 3.8 and Figure 3.9 show the evolution of the training and validation loss during the autoencoder-based training phase, where the model was successively trained on OFDM signals modulated with QPSK, 64QAM, and 1024QAM.

In the first figure, the training and validation loss values are presented in their natural scale. As observed, the loss rapidly decreases during the early epochs, especially during the QPSK phase (first one), indicating that the model quickly captures the fundamental structure of the lower-complexity modulation. As training progresses through the more complex modulation schemes (64QAM and 1024QAM), the loss continues to decrease steadily, although with smaller margins, showing that the model is able to learn more intricate structures as well. It is worth noting how the loss temporarily increases at each modulation transition, suggesting that the model initially expects signal characteristics from the previous modulation scheme. However, it quickly adapts to the new modulation. Interestingly, the second transition (from 64QAM to 1024QAM) results in a smaller spike compared to the first, indicating that the model progressively internalizes generalizable features across modulation types. The vertical dotted lines indicate the boundary between training phases for each modulation type, with clear transitions visible in the loss behavior.

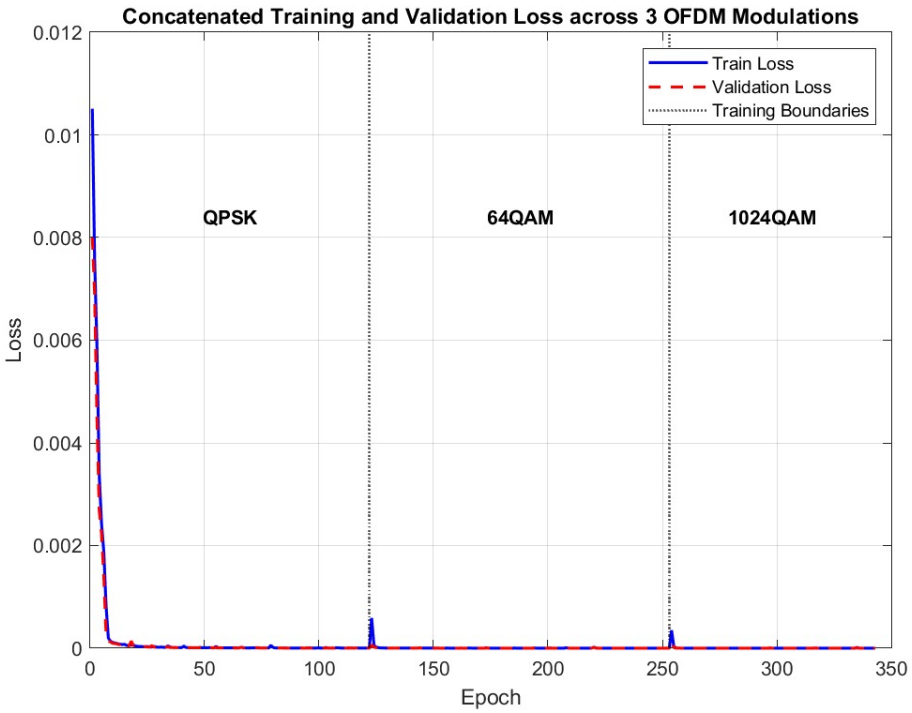


Fig. 3.8. Training and validation loss across the three consecutive training phases with OFDM signals (QPSK, 64QAM, and 1024QAM).

To better appreciate the progression of the loss, particularly in the lower ranges, Figure 3.9 shows the same curves on a logarithmic scale. This representation reveals the fine-scale convergence behavior of the network and highlights the periods of minor overfitting or instability, which appear as small spikes. The log-scale plot confirms that even for the highest-order modulation (1024QAM), the loss eventually stabilizes to very low values ($< 10^{-6}$), indicating that the model successfully reconstructs the signals in the autoencoder setting regardless of modulation complexity.

These results demonstrate that the model can adapt to signals of increasing complexity, from QPSK to 1024-QAM, and learn general features of OFDM signals. As more complex modulations were introduced, the model continued to reduce loss, albeit at a slower rate, indicating ongoing learning. This progress prepares the model for more challenging tasks like interference mitigation. The low final MSE values suggest that the learned latent representation is effective and generalizable, setting the stage for the next training phase focused on interference mitigation.

Now, in order to assess the generalization capabilities of the trained model beyond the specific configurations used during training, we conducted several inference tests using the MATLAB demodulation and visualization application described previously. For this purpose, we evaluated the model on a variety of OFDM signals modulated with schemes that were either not used during training or altered in terms of their modulation parameters. Table 3.2 summarizes the Bit Error Rate (BER) along with the MSE results obtained in each scenario.

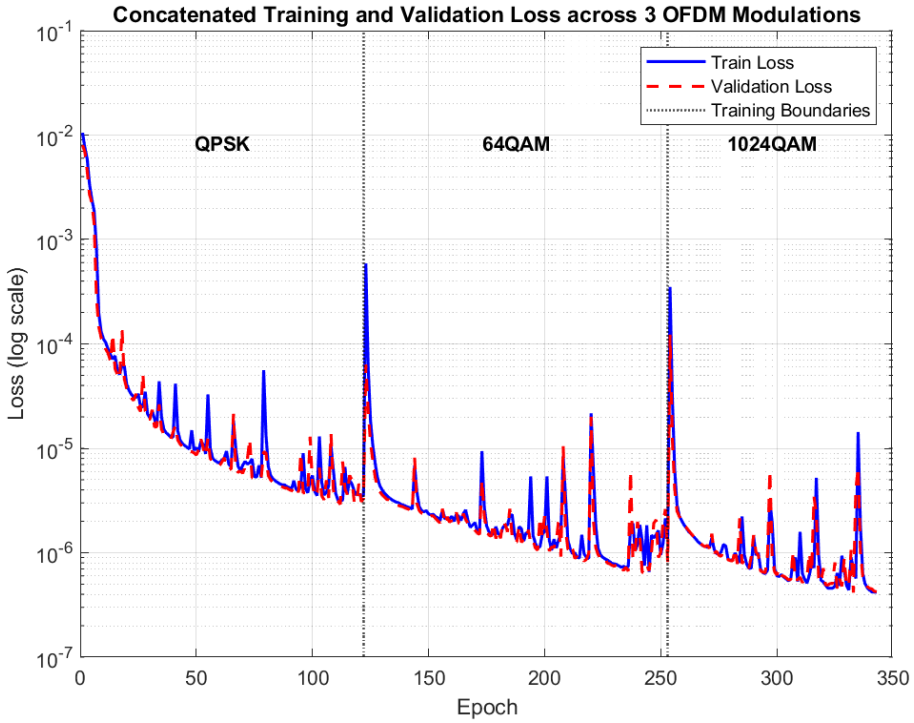


Fig. 3.9. Training and validation loss plotted on logarithmic scale, showing improved detail of convergence behavior.

Modulation Type	**Configuration**	**MSE**	**BER**
OFDM BPSK	Original	$7.03 \cdot 10^{-7}$	0
OFDM QPSK	Original	$8.99 \cdot 10^{-7}$	0
OFDM 64QAM	Original	$7.40 \cdot 10^{-7}$	0
OFDM 256QAM	Original	$1.05 \cdot 10^{-6}$	0.0014
OFDM 1024QAM	Original	$4.11 \cdot 10^{-7}$	0.0221
OFDM QPSK	Modified Parameters	$1.22 \cdot 10^{-6}$	0
OFDM 64QAM	Modified Parameters	$2.00 \cdot 10^{-6}$	0
OFDM 256QAM	Modified Parameters	$2.42 \cdot 10^{-6}$	0
OFDM 1024QAM	Modified Parameters	$2.48 \cdot 10^{-6}$	0.0037
OFDM 64QAM	Original + Noise (SNR=10dB)	0.0013	0.3281

*Note. The reported MSE values in the table correspond to the **average mean squared error across all video sequences** for each modulation and configuration. In contrast, the BER values are computed from a **single representative video per modulation**, kept consistent across experiments for fair comparison.*

Table 3.2. Results on test OFDM Signals

As shown, the model demonstrates robust performance across a range of OFDM modulations, achieving zero BER for BPSK, QPSK, and 64QAM signals, even those not explicitly included in the training process. This aligns with their extremely low MSE values, typically below 10^{-6}, indicating that the model reliably reconstructs these signals with minimal deviation.

For higher-order modulations such as 256QAM and 1024QAM, the model still performs well, though both MSE and BER increase slightly. This is expected, as higher-order constellations have more closely spaced symbols, making them inherently more sensitive to interference and reconstruction errors. Notably, the BER for OFDM-1024QAM with original parameters is 0.0221, and the corresponding MSE remains low at $4.11 \cdot 10^{-7}$, suggesting that symbol-level errors may not always be reflected in large overall distortions. This could be expected in denser constellation modulations, such as this one, where symbols are closer to each other.

Interestingly, for the 256QAM signal using the original parameters, the visual result was remarkably clean despite the BER of 0.0014. The MSE in this case was $1.05 \cdot 10^{-6}$, reinforcing the idea that minor symbol-level errors may not significantly affect perceived quality—particularly when signal energy remains well-preserved.

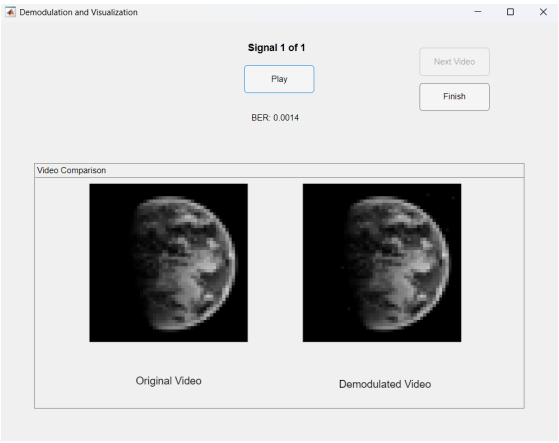
Furthermore, the results indicate that when using signals generated with modified modulation parameters, such as a shorter FFT length and fewer symbols, the model often performs even better—especially for 256QAM and 1024QAM. These configurations simplify the signal structure, making them easier for the model to interpret. This implies that the model has effectively learned the underlying structure of OFDM signals and is able to generalize across parameter variations to a considerable extent.

On the other hand, the test involving 64QAM signals with added noise (SNR = 10dB) reveals a notable degradation. While the average MSE is relateviley moderate (yet significantlly higher than for the other cases) at 0.0013, the BER spikes to 0.3281, confirming that the model is not equipped for denoising tasks. The MSE reflects some level of reconstruction consistency, but the high BER indicates that many symbols are misclassified, likely due to small shifts around decision boundaries. This is consistent with our design, where the training dataset did not include noisy signals.

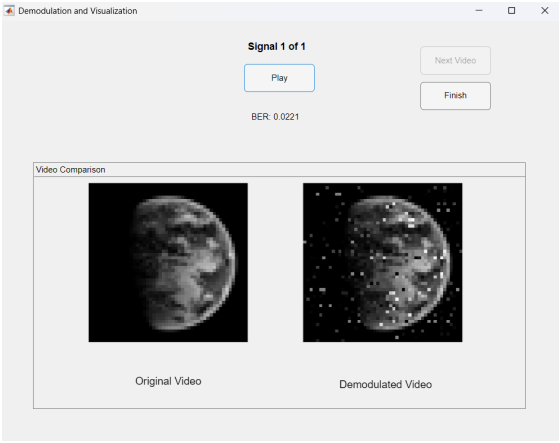
To complement the quantitative results, Figure 3.10 presents visual comparisons of five representative cases, highlighting the effects of different modulation types and configurations on the model's performance.

Stage 2: Interference Mitigation Training

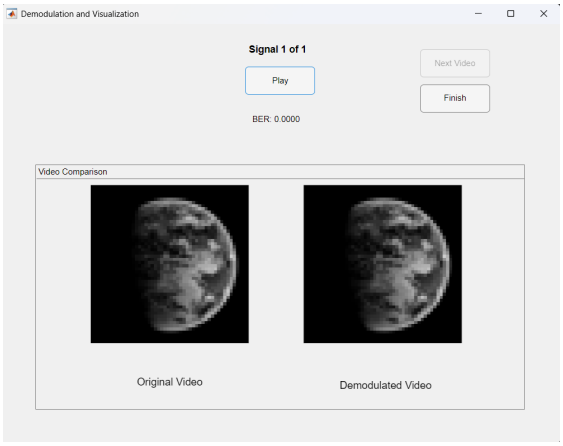
After validating the feasibility and effectiveness of the UNet model on clean signals in Stage 1, where the model learned to reconstruct undistorted OFDM signals, the second training phase introduces a significantly more complex challenge: the mitigation of interfering signals. For this purpose, the model trained in Stage 1, with its weights initialized



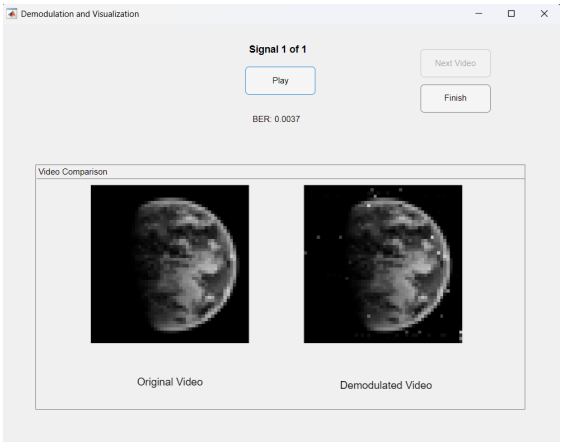
(a) OFDM 256QAM (BER:0.0014)



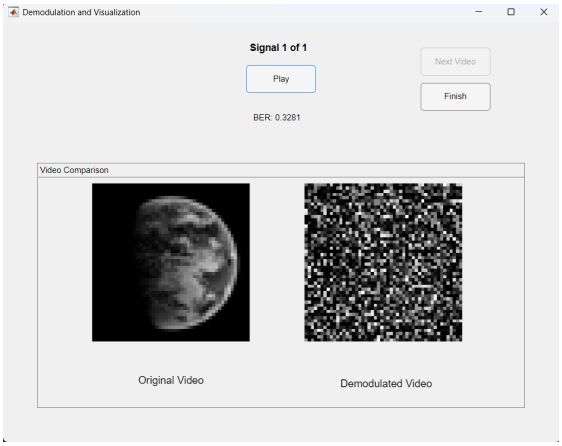
(b) OFDM 1024QAM (BER: 0.0221)



(c) OFDM 256QAM (modified) (BER:0.0000)



(d) OFDM 1024QAM (modified) (BER:0.0037)



(e) OFDM 64QAM + Noise (BER:0.3281)

Fig. 3.10. Visual comparison of inference results for various OFDM configurations and conditions after stage 1 of training (autoencoder).

based on the learned representations of clean OFDM signal structures, is fine-tuned to handle interference. The dedicated interference dataset, which contains OFDM signals of interest (SoI) corrupted with various types of interference (including other OFDM modulations and Bluetooth BR signals), is used in this stage. These interfered signals are processed with different attenuation factors, simulating varying degrees of interference strength, as detailed in the Interference Dataset Section (3.3.1). The transition from Stage 1 to Stage 2 is crucial, as the model builds upon the knowledge gained from clean signal recovery to adapt and learn to separate and recover the clean SoI under more challenging interference conditions.

Figure 3.11 and Figure 3.12 show the training and validation loss curves for this stage. Unlike the previous phase, the loss evolution extends across nearly 2900 epochs, highlighting the increased complexity of the task. This is expected, as not only are there much more samples and are the signals more diverse, but the goal of interference suppression requires disentangling the signal of interest from an overlapping source that can be both structurally similar (e.g., OFDM) or drastically different (e.g., Bluetooth).

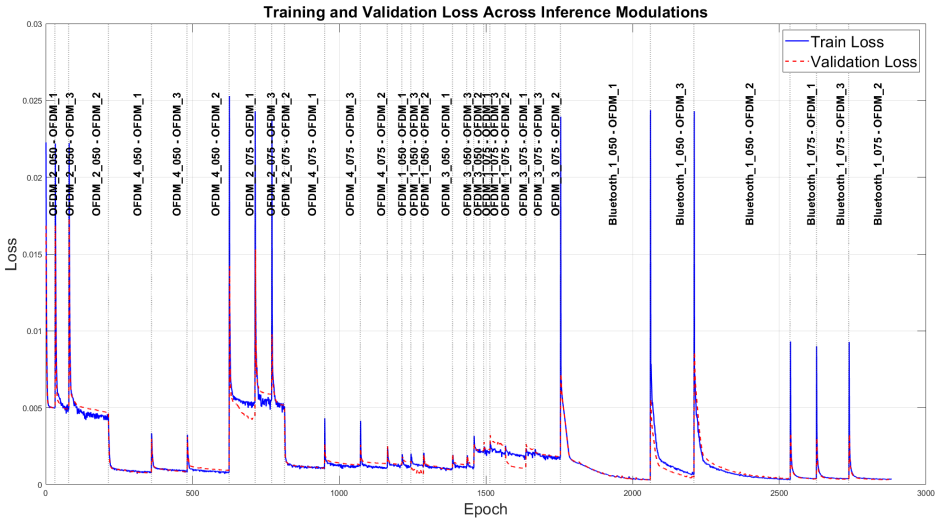


Fig. 3.11. Training and validation loss across all interference signals.

An interesting observation is that the training curves can be intuitively analyzed in segments of three, corresponding to the SoI modulation types (QPSK, 64QAM, and 1024QAM) trained together within each interference setup. This structural grouping is a direct result of the dataset organization, where each interference source and attenuation factor combination is paired with the three SoI types in separate subfolders.

The name labels over the segments follow the pattern `mod1_a_mod2`, where:

- `mod1` corresponds to the interfering signal modulation type.
- `a` is the attenuation factor applied to the interference (either 0.5 or 0.75, which would be 050 or 075, respectively).
- `mod2` represents the modulation used for the SoI.

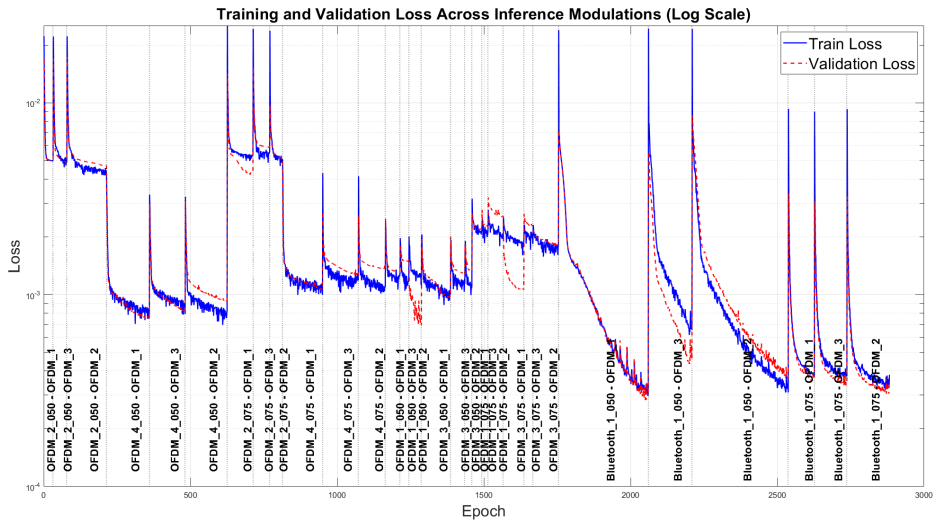


Fig. 3.12. Training and validation loss across all interference signals in logarithmic scales.

In our specific experiments:

- SoI modulations are mapped as: `OFDM1` → QPSK, `OFDM2` → 64QAM, `OFDM3` → 1024QAM.
- Interfering modulations include:
  - `OFDM1` → 1024QAM
  - `OFDM2` → 1024QAM with modified parameters
  - `OFDM3` → 256QAM
  - `OFDM4` → 256QAM with modified parameters
  - `Bluetooth1` → Bluetooth BR

It is also noticeable that when the attenuation factor is increased from 0.5 to 0.75, the loss tends to rise accordingly. This observation aligns with expectations, as a higher attenuation factor implies a stronger interference component, making the separation task more demanding for the model.

Another relevant point is the model's difficulty in reducing the loss during the later stages of training, particularly when dealing with Bluetooth BR interference. This is not surprising considering the structural disparity between Bluetooth signals and the OFDM-based SoIs seen during training, reinforcing the idea that generalization to cross-standard interference is considerably harder.

Moreover, the final loss values achieved in this phase are consistently higher than those observed during clean signal training. This again is coherent, as the task complexity is significantly elevated, and interference suppression poses greater challenges than pure denoising.

Additionally, we observe that SoI signals with higher-order modulation (such as 1024

QAM) tend to result in slightly higher losses compared to QPSK and 64QAM. This could indicate that the model learns to leverage the less dense constellation characteristics to better discriminate between useful and interfering components.

Once the model had been trained to handle a wide range of interference scenarios, we evaluated its performance on the test dataset, which included diverse combinations of interference types, attenuation factors, and signal of interest (SoI) modulations. Table 3.3 summarizes the results using two key metrics: the mean squared error (MSE) over the recovered signal and the bit error rate (BER) computed on a representative video over all video frames. BER values are presented both before and after applying the interference mitigation model. All image outputs corresponding to each entry in the table are included in Appendix A.1, which visually reinforce the numerical results discussed here.

Overall, the results reflect the complexity of the task. While the model is successful at suppressing interference in some specific cases (especially when the interfering signals are relatively weak or simple) it generally struggles to fully recover the clean signal in the presence of strong, high-order modulation interference. This is consistent with the expected difficulty of the task: high-order QAM modulations are more sensitive to distortion and overlap more in constellation space, which complicates the model's job in separating the SoI from the interference.

Among the positive observations, we note:

- **Strong mitigation for low attenuation factors (0.5):** In scenarios where the interference power is lower relative to the SoI, the model achieves near-zero BER for low-order modulations like QPSK. For instance, with OFDM 1024QAM interference and attenuation factor 0.5, the model eliminates almost all errors when the SoI is QPSK, as can be seen in Figure A.1a from Appendix A.1.
- **Improved performance with simpler SoIs:** Regardless of the interference type, QPSK consistently benefits the most from mitigation, which is expected due to its robustness and lower constellation density. This suggests the model is effective at recovering structurally simpler signals even in the presence of significant interference.
- **Impact of modified parameters:** Interfering signals with modified OFDM parameters appear to be easier to mitigate in some settings. This might be because the structural mismatch between the interfering signal and the training distribution of the SoI helps the model disentangle the two sources more effectively.

However, there are several limitations and challenges reflected in the results:

- **Limited mitigation for high-order SoIs:** In most configurations where the SoI is 1024QAM, the post-mitigation BER remains high. This can also be visualized in any of the Figures A.1c, A.1d, A.2a, A.2d, A.3a, A.3d, A.5a, A.5d. The complexity and fragility of such modulations under interference likely exceed the model's learned generalization capabilities.

Interference Type	Configuration	Att. Factor	SoI Type	MSE	Interference BER	Interference Mitigation BER
OFDM 1024QAM	Original	0.5	QPSK	0.00563	0.2079	0.0000
			64QAM	0.00549	0.2891	0.3582
			1024QAM	0.00541	0.4165	0.3769
		0.75	QPSK	0.00682	0.2643	0.1333
			64QAM	0.00664	0.3735	0.4006
			1024QAM	0.00655	0.4417	0.4230
	Modified Params	0.5	QPSK	0.01666	0.3640	0.2975
			64QAM	0.01655	0.4318	0.4489
			1024QAM	0.01654	0.4711	0.4610
		0.75	QPSK	0.03068	0.4056	0.3623
			64QAM	0.03054	0.4558	0.4672
			1024QAM	0.03057	0.4810	0.4748
OFDM 256QAM	Original	0.5	QPSK	0.00558	0.2043	0.0000
			64QAM	0.00542	0.2748	0.3527
			1024QAM	0.00541	0.4158	0.3687
		0.75	QPSK	0.00680	0.2580	0.1313
			64QAM	0.00654	0.3582	0.3936
			1024QAM	0.00655	0.4391	0.4225
	Modified Params	0.5	QPSK	0.00498	0.1733	0.0507
			64QAM	0.00491	0.1663	0.2934
			1024QAM	0.00487	0.3763	0.2789
		0.75	QPSK	0.00559	0.1790	0.0877
			64QAM	0.00553	0.2201	0.3081
			1024QAM	0.00549	0.3857	0.3265
Bluetooth BR	-	0.5	QPSK	0.00695	0.0075	0.2077
			64QAM	0.00690	0.3434	0.0576
			1024QAM	0.00692	0.1598	0.3884
		0.75	QPSK	0.00034	0.0686	0.1356
			64QAM	0.00033	0.2990	0.1508
			1024QAM	0.00033	0.2686	0.3624

*Note. The reported MSE values in the table correspond to the **average mean squared error across all video sequences** for each modulation and configuration. In contrast, the BER values are computed from a **single representative video per modulation**, kept consistent across experiments for fair comparison.*

Table 3.3. Interference Mitigation Results for Various Interference Signals

- **Bluetooth interference is particularly difficult:** Despite being structurally different from OFDM, the model has mixed success with Bluetooth interference. While MSE is sometimes low (especially with strong attenuation), the BER after mitigation often remains significant. This implies that while the signal may look visually clean, symbol errors still persist, highlighting the limitation of relying solely on pixel-domain losses during training. It is interesting though, the cases when 64QAM is used as the SoI (Figures A.5b, A.5e), where the BER significantly decreases, and the image is much more clearer. This results of interest because in all previous cases where the SoI was an OFDM signal of any kind, the best results where obtained for QPSK, as could be expected because of its constellation.
- **Attenuation factor plays a critical role:** As expected, increasing the attenuation factor from 0.5 to 0.75 makes the interference stronger, and the performance generally deteriorates. This is seen consistently across all interference types and SoI combinations.
- **Performance saturation in complex scenarios:** In many high-interference cases (e.g., OFDM 1024QAM with 0.75 attenuation), the post-mitigation BER remains close to that of the interference, suggesting that the model is unable to meaningfully recover the original signal under these difficult conditions. This may be due to the model overfitting to simpler interference scenarios during training, or to limitations in the capacity of the model architecture itself.

These results demonstrate that, although the model does exhibit generalization capabilities and can effectively mitigate interference in favorable settings, there remains significant room for improvement.

4. Discussion and Conclusions

4.1. Analysis of Resulting Dataset

The results obtained throughout the experimental section demonstrate the overall success of the generated dataset in supporting the development and evaluation of machine learning-based solutions for signal recovery under interference.

Despite relying on a relatively simple model architecture such as UNet, the model was able to learn effective representations and achieve signal reconstruction in a wide range of interference scenarios. This is particularly encouraging given the complexity of the RF environment and the diversity of modulations, signal configurations, and interference types tested.

One of the key strengths of the dataset lies in its flexibility and scalability. The MATLAB- and Python-based functions developed for generating both clean and interfered signals allow for rapid expansion and reproducibility of experiments. Additionally, the Python classes implemented to load and process the HDF5-ormatted datasets, both in single-signal and interference modes, facilitate seamless integration into machine learning workflows, enabling consistent data access and preprocessing.

The dataset's unique design stands out due to the generation of signals based on real video content. Unlike traditional datasets that rely on randomly generated bitstreams, this dataset uses videos as a source to encode the signals, which adds a layer of realism and complexity to the data. This approach offers a significant advantage: the signals generated from video content are more structured, allowing models to learn temporal and spatial relationships inherent in the video data. By doing so, the signals obtained for training and testing have a higher degree of richness, enabling more advanced machine learning applications. This unique aspect not only provides a more accurate representation of real-world signals but also adds an extra dimension to the model's interpretation, which is not typically found in other datasets. This video-based approach opens the door to applications such as semantic communications, where the understanding of content is crucial for recovery, making this dataset particularly valuable for tasks requiring higher-level understanding of transmitted information.

Additionally, the modular nature of the signal generation process further enhances the dataset's flexibility. The system is designed in such a way that components involved in signal generation, such as the video encoding or modulation blocks, can be replaced or adjusted easily. This means that if other types of content, such as audio or text, need to be encoded into signals, the corresponding blocks can be swapped out without altering the core structure of the signal generation pipeline. This feature ensures that the dataset can be adapted to new use cases, providing both scalability and versatility for future research and model development.

Overall, the dataset has proven to be a robust and versatile tool for prototyping RF

signal restoration tasks with machine learning, offering both high fidelity and structured complexity. These results lay the groundwork for further exploration using more advanced model architectures, including transformer-based or foundation models, with the confidence that the data pipeline is reliable and informative.

4.2. Potential Impact of Dataset Generation on Future Research

The dataset developed in this work introduces a new paradigm in wireless dataset construction, especially suited for machine learning-based signal recovery and interference mitigation. Compared to existing benchmarks such as RadioML 2018.01A [2], WAIR-D [3], or RF Challenge datasets [1], our dataset offers several unique features that enhance its long-term research utility.

First, unlike datasets focused solely on classification tasks or limited to specific hardware and modulation scenarios, our dataset captures realistic structured transmissions embedded in video signals. This enables not only modulation recovery but also direct evaluation in terms of perceptual quality and application-layer impact (e.g., frame-level BER).

Second, the flexibility of the dataset generation framework, implemented in MATLAB and Python, allows researchers to simulate a wide range of interference types, power ratios, and signal configurations. This modularity ensures the dataset can be continuously extended to include emerging standards or more complex impairments, a limitation noted in other datasets such as Panoradio SDR [4] and DeepMIMO [6].

Third, by explicitly modeling cross-technology interference this dataset aligns more closely with practical coexistence challenges faced in dense wireless environments. Many existing datasets, including Radar [10] and WAIR-D[3], tend to focus on intra-technology scenarios or idealized signal conditions.

As such, this dataset provides a foundation not only for benchmarking denoising and source separation models, but also for the development of generalized wireless foundation models capable of operating across a spectrum of environments and technologies. This positions our dataset as a potential cornerstone for future open research efforts in RF signal understanding, intelligent receivers, and multi-technology communication systems.

4.3. Future Works and Developments Directions in the Data-Driven Framework for Wireless Signal Processing

The framework developed in this thesis is highly modular and adaptable, offering significant opportunities for future advancements in wireless signal processing. Given its design, any potential extensions or enhancements to the system are relatively straightforward, making it an ideal platform for evolving research and experimentation.

Future work could focus on expanding the capabilities of this framework in the following key areas:

- **Input Signal Diversity:** Support for new video formats (e.g., higher-resolution, variable frame rates) or non-video bitstreams (e.g., audio or textual data) can further diversify the dataset and help assess generalization across content types.
- **Technology Expansion:** The framework can be upgraded to include other wireless technologies such as 5G NR. This, along with many others, could be implemented, just like the ones already supported by the application, based on the *Wireless Waveform Generator* toolbox from MATLAB [19].
- **Demodulation Capabilities:** Incorporating WiFi demodulation and visualization functions would enable full end-to-end pipelines (from bitstream to video recovery) for these technologies (already implemented for modulation).
- **Real-World Capture Integration:** Adding capabilities for loading and augmenting real hardware captures (e.g., from SDRs) could enable domain adaptation and mixed-reality scenarios.
- **Channel State Information (CSI) Estimation:** The framework can be adapted to support Channel State Information (CSI) estimation, which is crucial for many modern wireless communication systems. With minimal modification, the current setup can incorporate techniques for estimating channel parameters, which would help improve the performance of communication systems, particularly in challenging environments with varying interference and signal degradation.
- **Foundation Models for Wireless Channels:** The current framework offers a solid foundation for the development of foundation models for wireless communication systems. These models aim to understand and generalize over different wireless channel behaviors and interference scenarios, similar to the approach in [34] that can learn generalized representations across diverse wireless technologies. By leveraging the rich and varied datasets, future versions of the framework could enable training of deep models capable of handling various wireless channel conditions, including dynamic changes in network load, environment, and interference types, improving robustness and adaptability.
- **Waveform Design and Optimization:** Thanks to the modularity of the framework, it can be adapted for waveform design, allowing for the optimization of transmitted signals based on specific communication tasks, content types, and environments. This includes tailoring waveforms for particular use cases (e.g., video or audio transmission) and enhancing robustness against interference. By leveraging machine learning, the framework can dynamically generate and optimize waveforms, improving communication performance in diverse and challenging wireless conditions.
- **Quality Assessment with Computer Vision Metrics:** Since, in the presented use case, we are using a ML model to recover (or clean) signals that correspond to videos, incorporating computer vision metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and object detection accuracy can

provide a more detailed evaluation of the recovered video quality, offering insights directly relevant to video transmission and restoration applications.

- **Interference Mitigation Specifics:** Building on the use case explored in this thesis, further work can be focused on improving interference mitigation strategies. Specifically, new models and techniques could be developed to better handle strong interference scenarios, such as those encountered in dense urban environments or with high-power interfering signals. The framework could be extended to allow more granular modeling of interference, including jamming detection, adaptive filtering, and real-time interference classification. By simulating a variety of interference types and signal distortions, we can enhance the effectiveness of machine learning models in real-world applications, making them more resilient to dynamic and complex interference conditions.

These advancements would push the boundaries of current wireless signal processing capabilities, enabling the development of more intelligent, adaptive, and efficient systems. The modularity and flexibility of the proposed framework ensure that it can easily evolve to meet the demands of future research and real-world communication systems.

Bibliography

[1] RFChallenge, *Icassp 2024 rf challenge: Data-driven signal separation in radio spectrum*, https://github.com/RFChallenge/icassp2024rfchallenge/blob/0.2.0/src/unet_model.py, Accessed: 2025-06-02, 2023.

[2] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[3] Y. Huangfu *et al.*, *Wair-d: Wireless ai research dataset*, 2022. arXiv: 2212.02159 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2212.02159.

[4] S. Scholl, *Classification of radio signals and hf transmission modes with deep learning*, 2019. arXiv: 1906.04459 [eess.SP]. [Online]. Available: https://arxiv.org/abs/1906.04459.

[5] S. Hanna, S. Karunaratne, and D. Cabric, *Wisig: A large-scale wifi signal dataset for receiver and channel agnostic rf fingerprinting*, 2022. arXiv: 2112.15363 [eess.SP]. [Online]. Available: https://arxiv.org/abs/2112.15363.

[6] A. Alkhateeb, *Deepmimo: A generic deep learning dataset for millimeter wave and massive mimo applications*, 2019. arXiv: 1902.06435 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1902.06435.

[7] M. J. Bocus *et al.*, *Operanet: A multimodal activity recognition dataset acquired from radio frequency and vision-based sensors*, 2021. arXiv: 2110.04239 [eess.SP]. [Online]. Available: https://arxiv.org/abs/2110.04239.

[8] A. Jagannath, Z. Kane, and J. Jagannath, *Bluetooth and wifi dataset for real world rf fingerprinting of commercial devices*, 2023. arXiv: 2303.13538 [cs.NI]. [Online]. Available: https://arxiv.org/abs/2303.13538.

[9] T. R. Oyedare, "A comprehensive analysis of deep learning for interference suppression, sample and model complexity in wireless systems," Ph.D. dissertation, Virginia Polytechnic Institute and State University, Arlington, Virginia, 2024. [Online]. Available: https://vtechworks.lib.vt.edu/items/4958b597-c261-406a-aa34-4b57a5e353cb.

[10] N.-C. Ristea, A. Anghel, and R. T. Ionescu, *Fully convolutional neural networks for automotive radar interference mitigation*, 2020. arXiv: 2007.11102 [eess.SP]. [Online]. Available: https://arxiv.org/abs/2007.11102.

[11] S. Ujan, N. Navidi, and R. J. Landry, "An efficient radio frequency interference (rfi) recognition and characterization using end-to-end transfer learning," *Applied Sciences*, vol. 10, no. 19, p. 6885, 2020. DOI: 10.3390/app10196885. [Online]. Available: https://www.mdpi.com/2076-3417/10/19/6885.

[12] *The CRAWDAD Project*, https://crawdad.org/, Accessed: 2025-05-28.

[13] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, IEEE, Jul. 2017. DOI: 10.1109/indin.2017.8104767. [Online]. Available: http://dx.doi.org/10.1109/indin.2017.8104767.

[14] *CRAWDAD: KyutechInterference Dataset*, https://ieee-dataport.org/open-access/crawdad-kyutechinterference, Accessed: 2025-05-28, 2025.

[15] *CRAWDAD: RutgersNoise Dataset*, https://ieee-dataport.org/open-access/crawdad-rutgersnoise, Accessed: 2025-05-28, 2025.

[16] S. Grimaldi, A. Mahmood, and M. Gidlund, *Real-time interference identification via supervised learning: Embedding coexistence awareness in iot devices*, 2018. arXiv: 1809.10085 [eess.SP]. [Online]. Available: https://arxiv.org/abs/1809.10085.

[17] MIT RF Challenge, *Rf challenge 2024 dataset*, Accedido: 2025-06-09, 2024. [Online]. Available: https://rfchallenge.mit.edu/datasets/.

[18] A. Lancho *et al.*, "Rf challenge: The data-driven radio frequency signal separation challenge," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 4083–4100, 2025. DOI: 10.1109/ojcoms.2025.3556319. [Online]. Available: http://dx.doi.org/10.1109/OJCOMS.2025.3556319.

[19] MathWorks, *Wireless waveform generator app*, https://es.mathworks.com/help/comm/ref/wirelesswaveformgenerator-app.html.

[20] A. F. Molisch, *Wireless Communications*, 2nd. IEEE Press, 2011. [Online]. Available: https://ieeexplore.ieee.org/book/5635423.

[21] J. Stephens and D. Norman, "Direct-sequence spread spectrum system," in *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference NAECON 1991*, 1991, 462–466 vol.1. DOI: 10.1109/NAECON.1991.165790.

[22] *Bluetooth core specification version 5.3*, Accessed: [Insert Date of Access Here], Bluetooth SIG, 2021. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-3/.

[23] *Ieee standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Higher speed physical layer (phy) extension in the 2.4 ghz band*, https://standards.ieee.org/ieee/802.11b/1166/, IEEE, 1999.

[24] *Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 5: Enhancements for higher throughput*, https://standards.ieee.org/ieee/802.11n/3952, IEEE, 2009.

[25] D.-J. Deng, K.-C. Chen, and R.-S. Cheng, "Ieee 802.11ax: Next generation wireless local area networks," in *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2014, pp. 77–82. DOI: 10.1109/QSHINE.2014.6928663.

[26] Panoradio SDR, *Overview of open datasets for rf signal classification*, https://panoradio-sdr.de/overview-of-open-datasets-for-rf-signal-classification/?utm_source=chatgpt.com, Accessed: 2025-05-20, 2025.

[27] *Vecteezy*, https://www.vecteezy.com/.

[28] MathWorks, *Wlanhesuconfig (wlan toolbox)*, https://es.mathworks.com/help/wlan/ref/wlanhesuconfig.html.

[29] *Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 ghz*, https://standards.ieee.org/ieee/802.11ac/4473, IEEE, 2013.

[30] MathWorks, *Wlanvhtconfig (wlan toolbox)*, https://es.mathworks.com/help/wlan/ref/wlanvhtconfig.html.

[31] MathWorks, *Wlannonhtconfig (wlan toolbox)*, https://es.mathworks.com/help/wlan/ref/wlannonhtconfig.html.

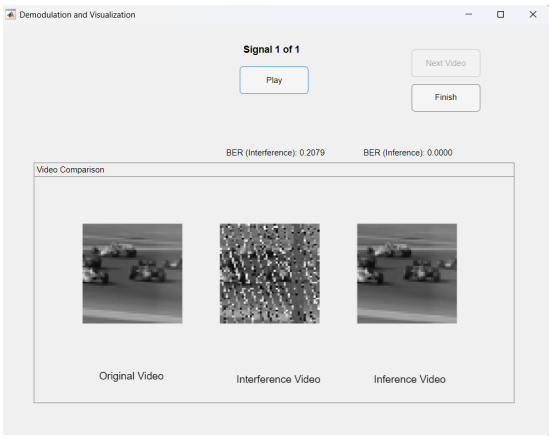
[32] M. Golbano, *Tfm foundation model for wireless signals*, https://github.com/mariogolbano/TFM-foundation-model-wireless-signals, Last updated: 2025-06-06, 2025.

[33] C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts*. Springer Cham, 2023. [Online]. Available: https://link.springer.com/book/10.1007/978-3-031-45468-4.

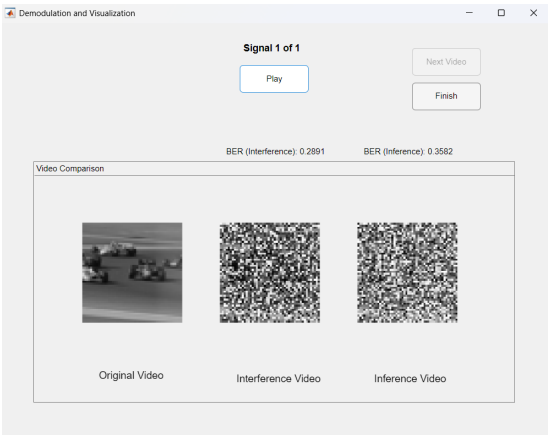
[34] S. Alikhani, G. Charan, and A. Alkhateeb, *Large wireless model (lwm): A foundation model for wireless channels*, https://lwm-wireless.net/, Preprint available at arXiv:2411.08872, Nov. 2024.

A. Appendices

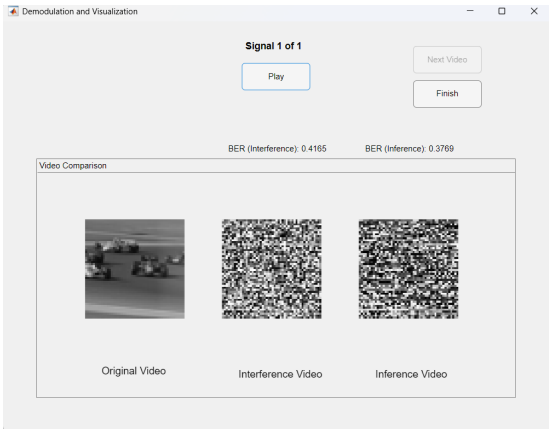
A.1. Appendix I: Visual Results of Interference Mitigation



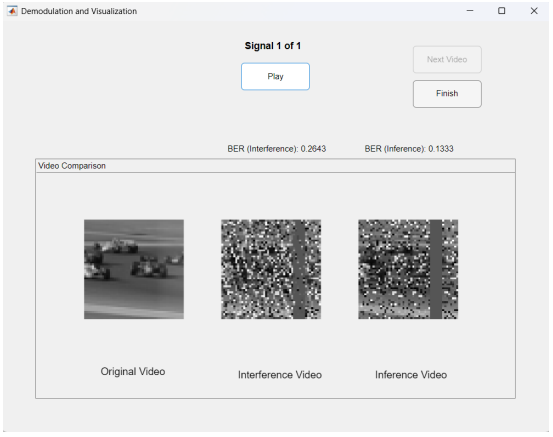
(a) OFDM 1024QAM, Att. 0.5, SoI: QPSK



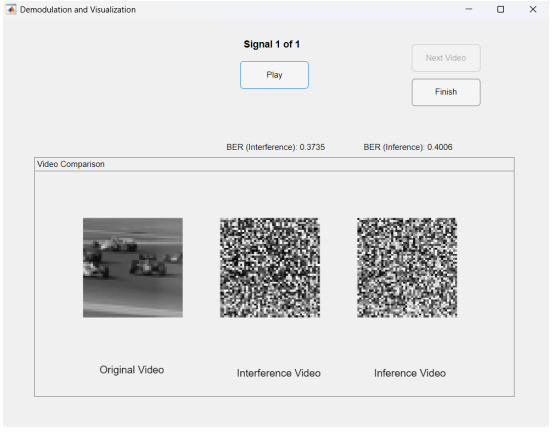
(b) OFDM 1024QAM, Att. 0.5, SoI: 64QAM



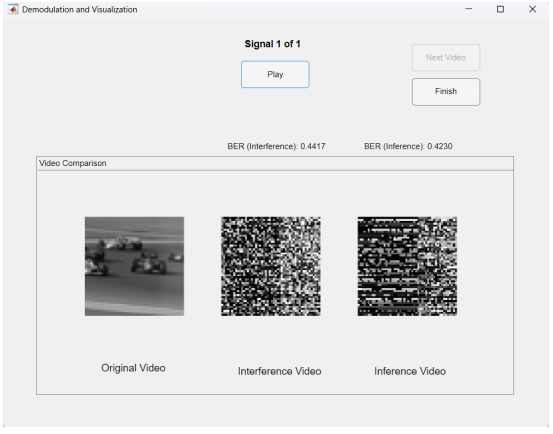
(c) OFDM 1024QAM, Att. 0.5, SoI: 1024QAM



(d) OFDM 1024QAM, Att. 0.75, SoI: QPSK

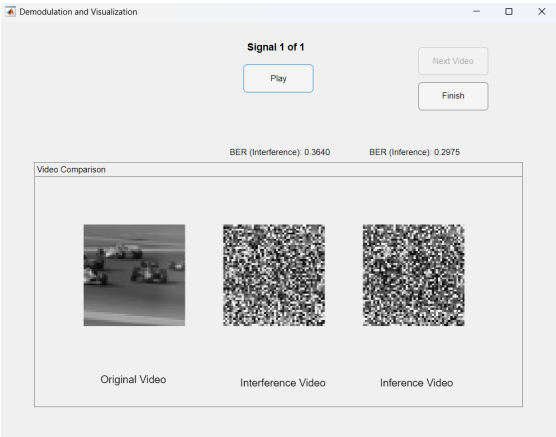


(e) OFDM 1024QAM, Att. 0.75, SoI: 64QAM

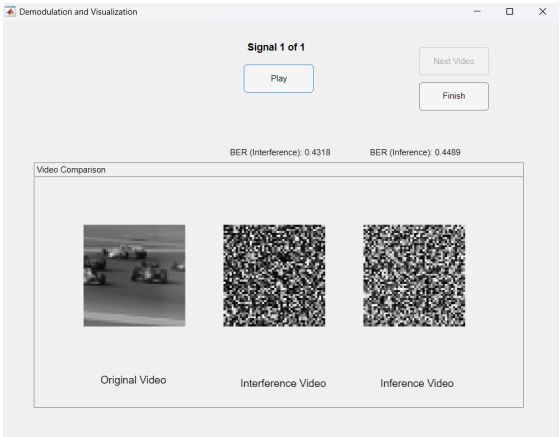


(f) OFDM 1024QAM, Att. 0.75, SoI: 1024QAM

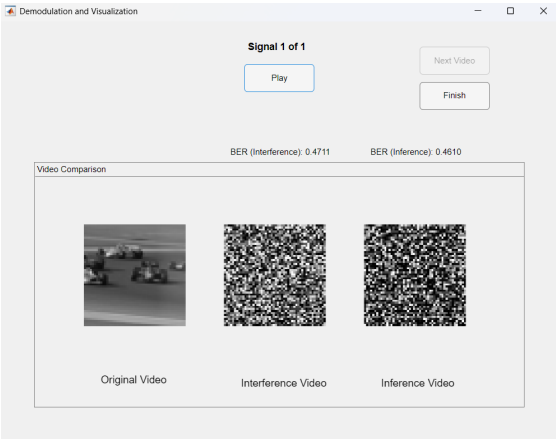
Fig. A.1. Recovered video frames after interference mitigation across different interference types, attenuation factors, and signal of interest (SoI) modulations (I).



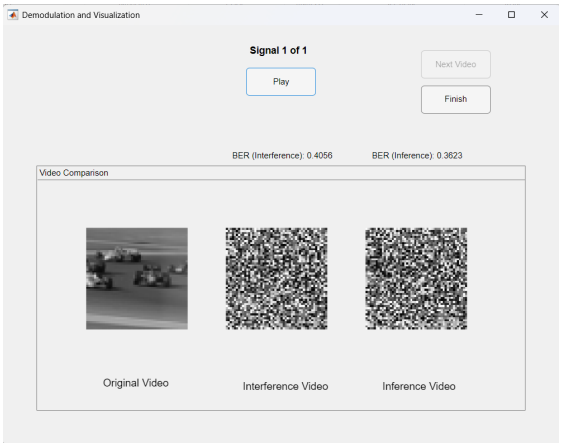
(a) OFDM 1024QAM (mod), Att. 0.5, SoI: QPSK



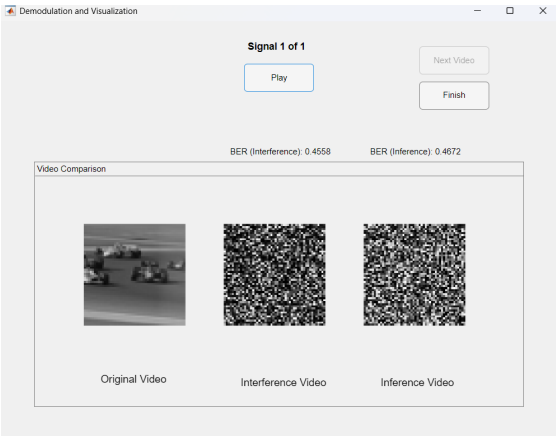
(b) OFDM 1024QAM (mod), Att. 0.5, SoI: 64QAM



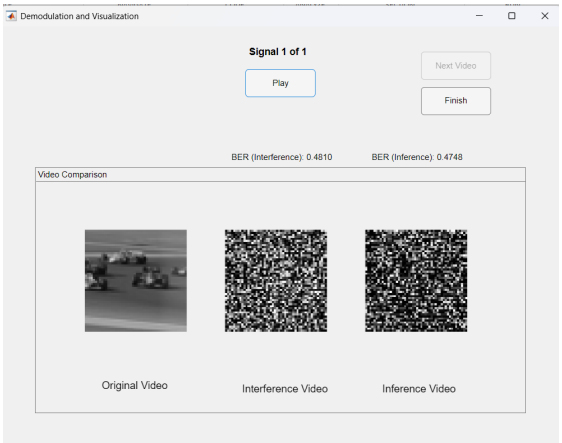
(c) OFDM 1024QAM (mod), Att. 0.5, SoI: 1024QAM



(d) OFDM 1024QAM (mod), Att. 0.75, SoI: QPSK

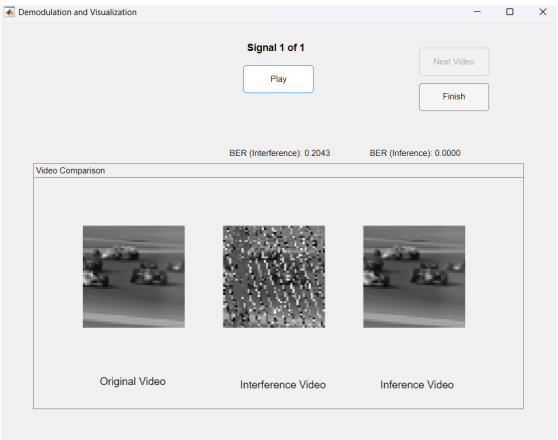


(e) OFDM 1024QAM (mod), Att. 0.75, SoI: 64QAM

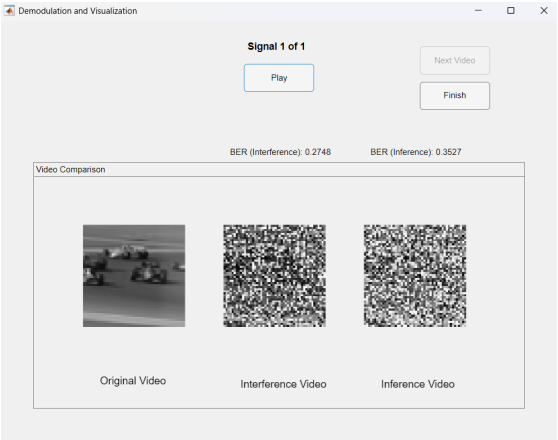


(f) OFDM 1024QAM (mod), Att. 0.75, SoI: 1024QAM

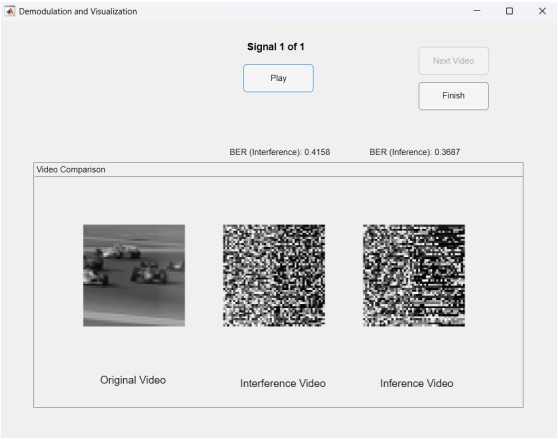
Fig. A.2. Recovered video frames after interference mitigation across different interference types, attenuation factors, and signal of interest (SoI) modulations (II).



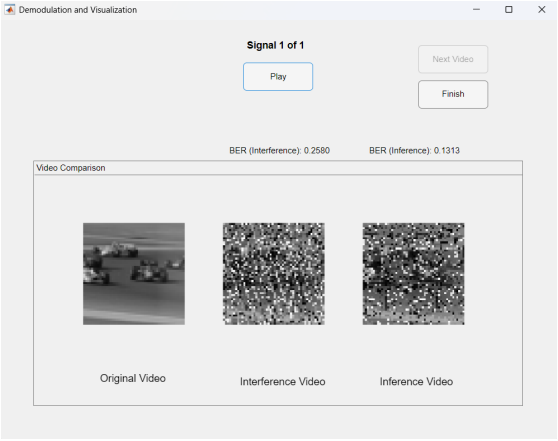
(a) OFDM 256QAM, Att. 0.5, SoI: QPSK



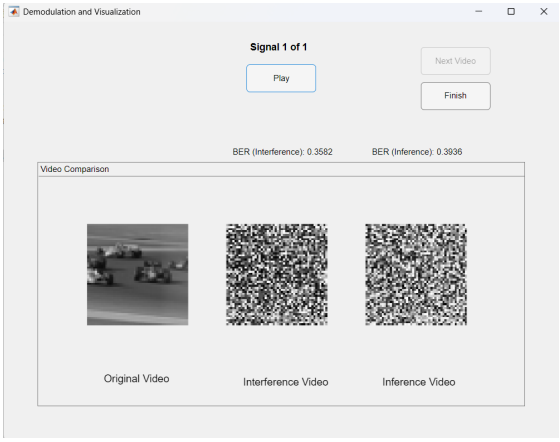
(b) OFDM 256QAM, Att. 0.5, SoI: 64QAM



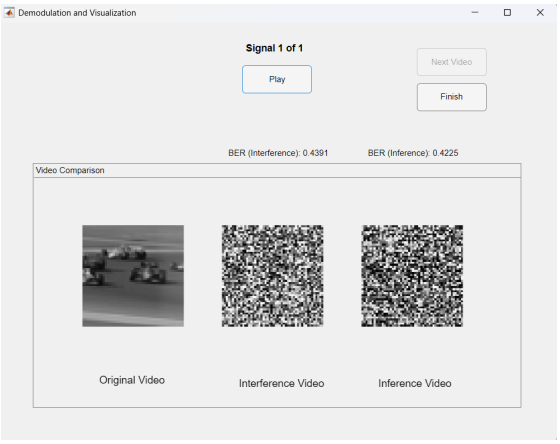
(c) OFDM 256QAM, Att. 0.5, SoI: 1024QAM



(d) OFDM 256QAM, Att. 0.75, SoI: QPSK

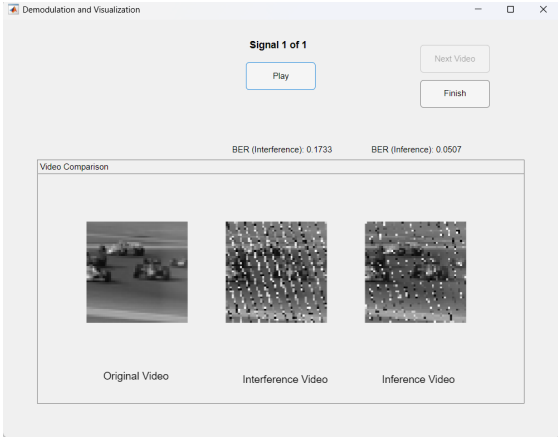


(e) OFDM 256QAM, Att. 0.75, SoI: 64QAM

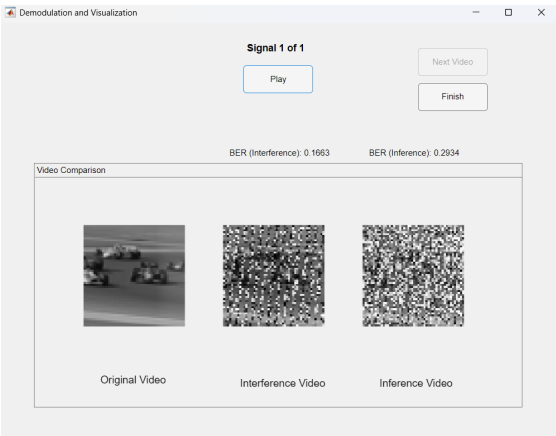


(f) OFDM 256QAM, Att. 0.75, SoI: 1024QAM

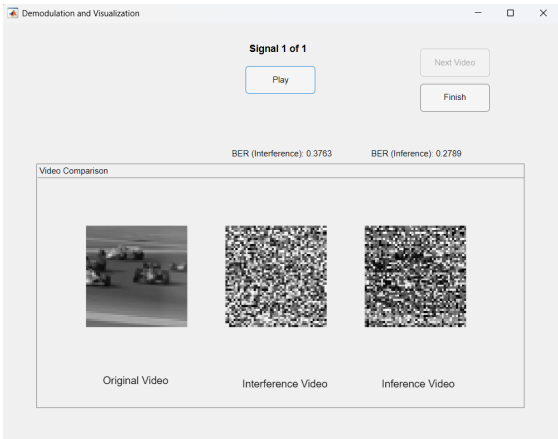
Fig. A.3. Recovered video frames after interference mitigation across different interference types, attenuation factors, and signal of interest (SoI) modulations (III).



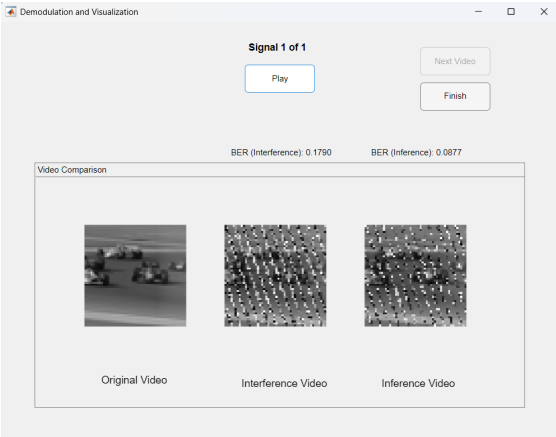
(a) OFDM 256QAM (mod), Att. 0.5, SoI: QPSK



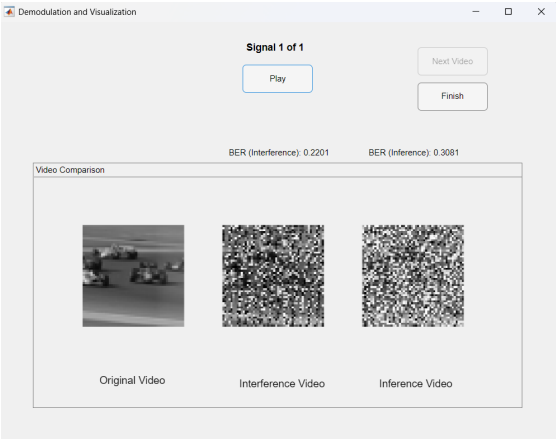
(b) OFDM 256QAM (mod), Att. 0.5, SoI: 64QAM



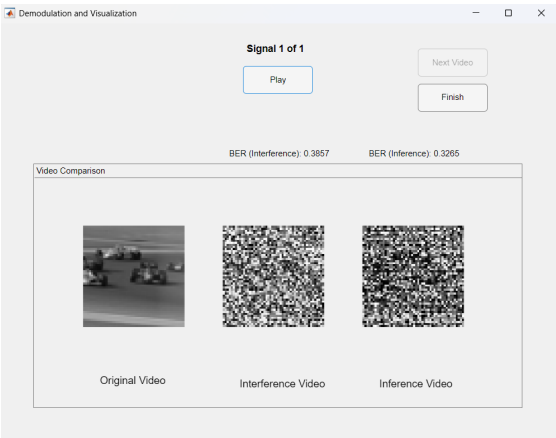
(c) OFDM 256QAM (mod), Att. 0.5, SoI: 1024QAM



(d) OFDM 256QAM (mod), Att. 0.75, SoI: QPSK

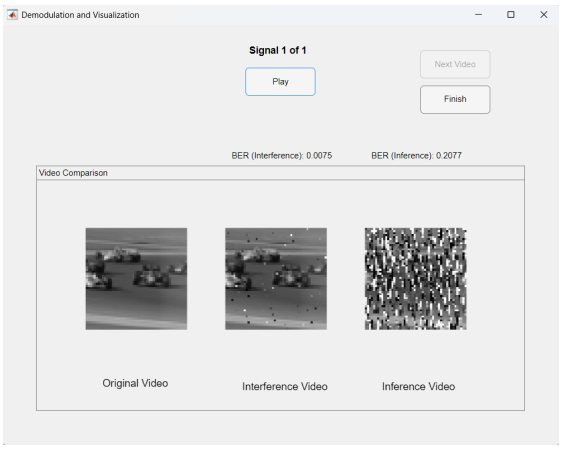


(e) OFDM 256QAM (mod), Att. 0.75, SoI: 64QAM

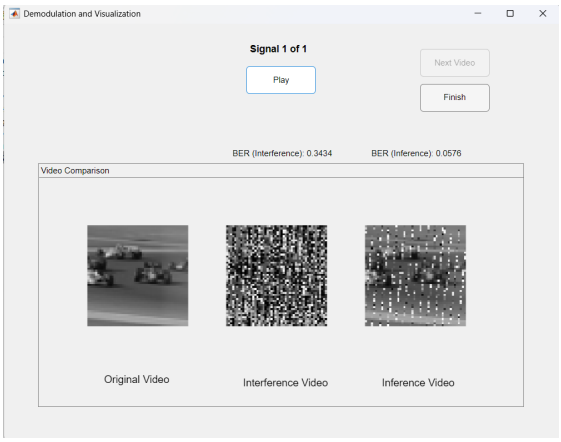


(f) OFDM 256QAM (mod), Att. 0.75, SoI: 1024QAM

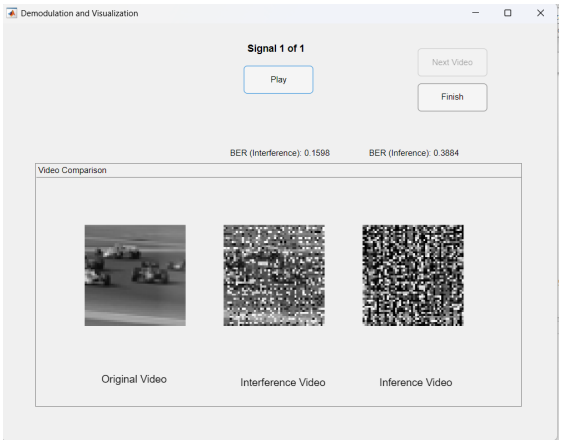
Fig. A.4. Recovered video frames after interference mitigation across different interference types, attenuation factors, and signal of interest (SoI) modulations (IV).



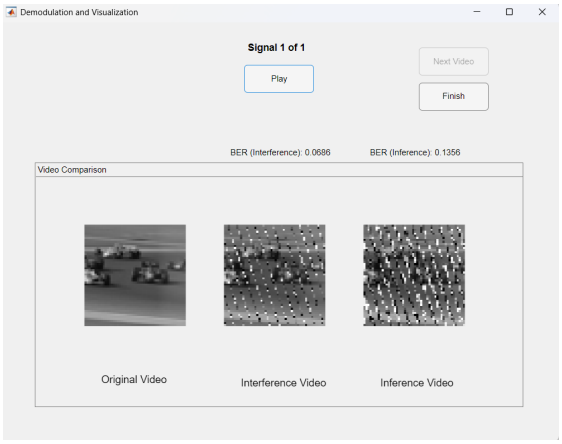
(a) Bluetooth BR, Att. 0.5, SoI: QPSK



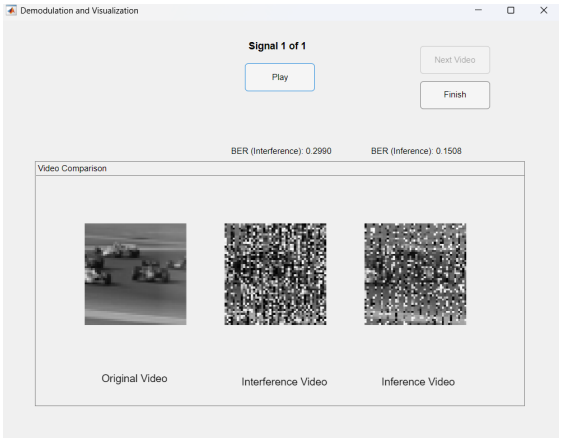
(b) Bluetooth BR, Att. 0.5, SoI: 64QAM



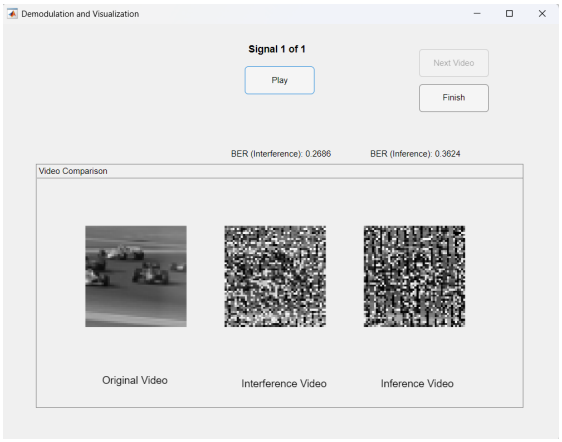
(c) Bluetooth BR, Att. 0.5, SoI: 1024QAM



(d) Bluetooth BR, Att. 0.75, SoI: QPSK



(e) Bluetooth BR, Att. 0.75, SoI: 64QAM



(f) Bluetooth BR, Att. 0.75, SoI: 1024QAM

Fig. A.5. Recovered video frames after interference mitigation across different interference types, attenuation factors, and signal of interest (SoI) modulations. (V)

uc3m | Universidad Carlos III de Madrid

DECLARATION OF USE OF GENERATIVE IA IN THE MASTER FINAL PROJECT

I have used Generative AI in this work

Check all that apply:

YES	NO
X	

If you have ticked YES, please complete the following 3 parts of this document:

Part 1: Reflection on ethical and responsible behaviour

Please be aware that the use of Generative AI carries some risks and may generate a series of consequences that affect the moral integrity of your performance with it. Therefore, we ask you to answer the following questions honestly (*please tick all that apply*):

Question		
In my interaction with Generative AI tools, I have submitted **sensitive data** with the consent of the data subjects.		
YES, I have used this data with permission	NO, I have used this data without authorisation	NO, I have not used sensitive data X
In my interaction with Generative AI tools, I have submitted **copyrighted materials** with the permission of those concerned.		
YES, I have used these materials with permission	NO, I have used these materials without permission	NO, I have not used protected materials X
In my interaction with Generative AI tools, I have submitted **personal data** with the consent of the data subjects.		
YES, I have used this data with permission	NO, I have used this data without authorisation	NO, I have not used personal data X

My use of the Generative AI tool has **respected its terms of use**, as well as the essential ethical principles, not being maliciously oriented to obtain an inappropriate result for the work presented, that is to say, one that produces an impression or knowledge contrary to the reality of the results obtained, that supplants my own work or that could harm people.	
YES	NO

If you **did NOT** have the permission of those concerned in any of questions 1, 2 or 3, briefly explain why *(e.g. "the materials were protected but permitted use for this purpose" or "the terms of use, which can be found at this address (...), prevent the use I have made, but it was essential given the nature of the work".*

Part 2: Declaration of technical use

Use the following model statement as many times as necessary, in order to reflect all types of iteration you have had with Generative AI tools. Include one example for each type of use where indicated: *[Add an example].*

I declare that I have made use of the Generative AI system ChatGPT for:

Develop specific content

Generative AI has been used as a support tool for the development of the specific content of the dissertation (MFP), including:

- *Assistance in the development of lines of code (programming). Since the based Macihne Learning model I used was based on tensorflow, and I considered more fit the use of pytorch, I used it as a help for carrying out these changes.*

For example: I have asked to help me change and adapt the UNet model from tensorflow to pytorch.

- *Development of GUI in MATLAB*

For example: I requested help to generate MATLAB code for a GUI that displays and processes image data interactively, including buttons, axes, and input fields.

Part 3: Reflection on utility

Please provide a personal assessment (free format) of the strengths and weaknesses you have identified in the use of Generative AI tools in the development of your work. Mention if it has helped you in the learning process, or in the development or drawing conclusions from your work.

Using Generative AI tools in my thesis was really helpful, especially when building MATLAB GUIs. I had not worked with MATLAB for graphical interfaces before, and the AI made it easier to test ideas and write interface code more quickly, saving time and allowing me to focus on design and user interaction. *As I also mentioned, ChatGPT helped me convert the UNet model from TensorFlow to PyTorch, which I preferred. Although this wasn't a central part of the thesis, it saved me time, as it could have been a tedious process. That said, not everything was ideal. Sometimes the code suggestions weren't optimal or fully accurate, so I had to double-check, debug, and even research certain aspects myself. However, that process helped me gain a deeper understanding of both GUI development and the UNet model, making the experience doubly beneficial: I learned more while also saving valuable time.*