InterfO-RAN: Real-Time In-band Cellular Uplink Interference Detection with GPU-Accelerated dApps

Neagin Neasamoni Santhi, Davide Villa, Michele Polese, Tommaso Melodia
Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A
{neasamonisanthi.n, villa.d, m.polese, t.melodia}@northeastern.edu

Abstract

Ultra-dense fifth generation (5G) and beyond networks leverage spectrum sharing and frequency reuse to enhance throughput, but face unpredictable in-band uplink (UL) interference challenges that significantly degrade Signal to Interference plus Noise Ratio (SINR) at affected Next Generation Node Bases (gNBs). This is particularly problematic at cell edges, where overlapping regions force User Equipments (UEs) to increase transmit power, and in directional millimeter wave systems, where beamforming sidelobes can create unexpected interference. The resulting signal degradation disrupts protocol operations, including scheduling and resource allocation, by distorting quality indicators like Reference Signal Received Power (RSRP) and Received Signal Strength Indicator (RSSI), and can compromise critical functions such as channel state reporting and Hybrid Automatic Repeat Request (HARQ) acknowledgments.

To address this problem, this article introduces InterfO-RAN, a real-time programmable solution that leverages a Convolutional Neural Network (CNN) to process In-phase and Quadrature (I/Q) samples in the gNB physical layer, detecting in-band interference with accuracy exceeding 91% in under 650 μs. InterfO-RAN represents the first O-RAN dApp accelerated on Graphics Processing Unit (GPU), coexisting with the 5G NR physical layer processing of NVIDIA Aerial. Deployed in an end-to-end private 5G network with commercial Radio Units (RUs) and smartphones, our solution was trained and tested on more than 7 million NR UL slots collected from real-world environments, demonstrating robust interference detection capabilities essential for maintaining network performance in dense deployments.

CCS Concepts

• **Networks** → **Network performance analysis**.

Keywords

dApp, 5G NR, O-RAN, CUDA-Accelerated RAN, Inference, GPU

ACM Reference Format:
Neagin Neasamoni Santhi, Davide Villa, Michele Polese, Tommaso Melodia. 2025. InterfO-RAN: Real-Time In-band Cellular Uplink Interference Detection with GPU-Accelerated dApps. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '25), October 27–30, 2025, Houston, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3704413.3764454

MobiHoc '25, October 27–30, 2025, Houston, TX, USA

ACM ISBN 979-8-4007-1353-8/25/10
https://doi.org/10.1145/3704413.3764454

1 Introduction

The evolution of wireless communication networks has been driven by the increasing demand for high-speed, low-latency, and ultra-reliable connectivity to support emerging applications such as autonomous systems, industrial automation, and immersive experiences. To meet these demands, 5G-and-beyond networks leverage massive densification to improve throughput and latency within the limited spectrum allocated to mobile communications [1]. In ultra-dense networks, multiple neighboring small cells are configured to reuse the same frequency bands to increase frequency reuse [2]. Directional millimeter wave systems further push this concept, with commercial deployments often leveraging the same 400 MHz or 800 MHz bands at 28 GHz and 39 GHz for all base stations [3]. Finally, private network deployments (e.g., for enterprise scenarios) use limited portions of shared spectrum, with all deployments constrained in the same 100 MHz (e.g., Germany, Brazil, Netherlands) or 150 MHz (e.g., U.S., with CBRS) [4].

The Need for Real-Time Interference Detection. However, densification, sharing, and spectrum reuse also increase inter-cell interference. Although significant research has focused on coordination mechanisms to reduce interference in downlink [5–8], in-band uplink (UL) interference remains a challenging problem, especially considering the unpredictability of user mobility and configurations and thus of the source of interference. This occurs when unwanted transmissions from User Equipments (UEs), operating within the same frequency band but connected to different Next Generation Node Bases (gNBs), interfere with the UL reception of a serving gNB, as shown in Fig. 1. Such an overlap can significantly degrade the UL Signal to Interference plus Noise Ratio (SINR) at the affected gNB, potentially leading to unrecoverable packets and thus reduced network performance [9]. This is particularly significant when UEs are located at the cell edge, where overlapping regions between two neighboring cells force the devices to increase their transmit power to overcome the link budget constraints. In such cases, the

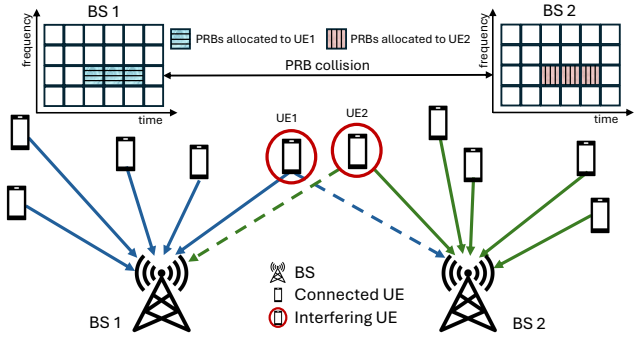


Figure 1: In-band UL interference overview (dashed lines).

unpredictability of potential in-band UL interference is closely associated with UE power control. Similarly, UEs may transmit at excessive power levels to compensate for signal degradation when experiencing Non-Line-of-Sight (NLOS) conditions with respect to their serving node. While this adjustment helps maintain reliable links, it can also lead to unintended interference in adjacent cells. Moreover, in scenarios involving directional UE transmissions, as in millimeter wave links, beamforming introduces an additional source of unpredictability. It can impact adjacent cells through strong sidelobes, or, under highly dynamic conditions, even affect them with the main lobe. These conditions are often unknown apriori (e.g., at scheduling time), and can vary on timescales faster than a subframe (i.e., sub-ms). This calls for real-time interference detection to enable a prompt reaction across the protocol stack.

The impact of in-band UL interference extends beyond immediate signal degradation, and can disrupt protocol operations like scheduling and resource allocation at the gNB. To execute these operations, the gNB relies on quality indicators like Reference Signal Received Power (RSRP) and Received Signal Strength Indicator (RSSI). However, interference distorts the gNB's assessment of UE channel conditions, introducing errors that propagate across multiple slots and affect both UL and Downlink (DL) transmissions. This disruption becomes particularly critical when interference affects control signaling (e.g., Uplink Control Indication (UCI)) or DeModulation Reference Signal (DMRS) used for channel estimation. Because of the inherently lower power of UL signals compared to DL transmissions, even interference at low power levels can significantly impair communication quality. This forces the gNB to allocate additional resources through retransmissions and more robust coding schemes, thereby reducing spectral efficiency and limiting overall user capacity. In scenarios with severe interference, connection drops can occur, underscoring the importance of developing and implementing robust interference detection and mitigation methods in practical deployments.

Contribution: InterfO-RAN. In this paper, we address the critical challenge of UL **interference detection and mitigation** by presenting InterfO-RAN, a novel real-time programmable solution that integrates a Convolutional Neural Network (CNN) within the gNB physical layer to process and analyze In-phase and Quadrature (I/Q) samples. Our solution achieves interference detection with accuracy exceeding 91% in less than 650 μs, enabling practical implementation in production cellular networks.

Designed as a pluggable, programmable component, InterfO-RAN is the first to extend Graphics Processing Unit (GPU) acceleration to dApps for real-time CNN-based interference detection. The InterfO-RAN dApp is integrated with the NVIDIA Aerial 5G NR Physical (PHY) layer following the emerging O-RAN dApp paradigm [10]. This represents a fundamental system-level innovation, enabling pluggable Artificial Intelligence (AI)/Machine Learning (ML) models for 5G and beyond and accelerated on GPU, aligning with AI-for-RAN use cases within the AI-RAN Alliance [11]. InterfO-RAN's design requires GPU resource management, seamless interfacing between the dApp inference (using ONNX Runtime (ORT)) and the PHY, and careful model design (quantized, minimal, yet effective CNN). The dApp accesses UL data (I/Q samples and channel quality Key Performance Indicators (KPIs)) and telemetry

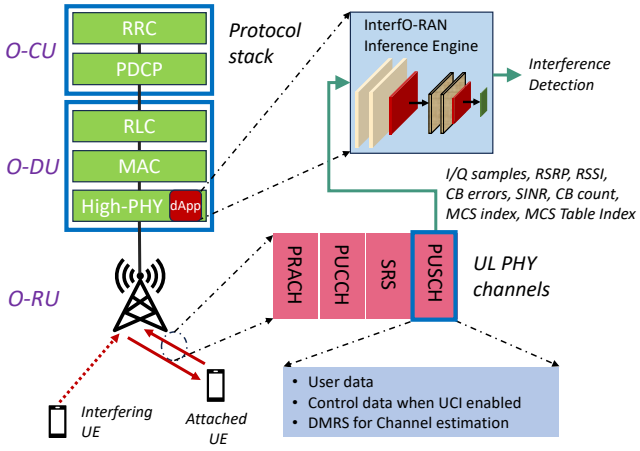


Figure 2: System Architecture.

and provides a reusable, extensible interface for additional real-time ML-based functionalities in the O-RAN stack. The same dApp, indeed, can be repurposed for tasks like Angle of Arrival (AoA) Estimation or 5G positioning.

InterfO-RAN is implemented as a set of pipelines to expose necessary information from Aerial Compute Unified Device Architecture (CUDA) kernels related to the Physical Uplink Shared Channel (PUSCH), as well as additional dApp CUDA kernels that implement CNN with ORT. We selected CNNs as they effectively recognize patterns in complex I/Q planes. We deploy InterfO-RAN and the Aerial stack in an end-to-end private 5G network. The private 5G network is a multi-vendor, fully virtualized, and GPU-accelerated production network with multiple commercial Radio Units (RUs) and smartphones [12], where the higher layers of the 5G stack are implemented with OpenAirInterface (OAI). We leverage this real-world setup to collect data with and without interference from a deployment spanning two different buildings. More than seven million NR UL slots, along with artificially generated data in MATLAB, are used to train and test eight configurations of the CNN, employing Transfer Learning (TL) techniques as well. The selected solution is then evaluated Over-the-Air (OTA) across various scenarios with different pairs of interfering RUs. Our results demonstrate the robustness of InterfO-RAN in detecting interference, achieving over 91% accuracy in less than 650 μs, while imposing minimal strain on PHY operations.

The remainder of the paper is organized as follows. Section 2 presents the system architecture overview. Section 3 provides some background on the Aerial framework. Section 4 describes the system design and implementation of the InterfO-RAN dApp, including details on the CNN and data pipelines. Section 5 outlines the experimental setup and data collection operations, while Sec. 6 discusses the results. Related work is reviewed in Sec. 7. Finally, Sec. 8 concludes the paper and outlines directions for future research.

2 System Architecture

O-RAN Primer. The system model of this paper develops on top of the O-RAN architecture. O-RAN disaggregates the traditional monolithic gNB into modular, programmable components using open interfaces, i.e., the RU, Distributed Unit (DU), and Central Unit (CU), as shown in Fig. 2. Within this framework, the features and

capabilities of the Radio Access Network (RAN) can be extended with custom logic that interfaces with the disaggregated gNB, implementing control loops for inference, control, and classifications. Among these, dApps are distributed applications designed to bring intelligence to CUs and DUs, supporting real-time inference and control loops operating at sub-10 ms timescales [10]. dApps complement xApps and rApps residing within external RAN Intelligent Controllers (RICs) [13].

The InterfO-RAN Framework. Figure 2 illustrates our system architecture, where InterfO-RAN is integrated as a programmable dApp into the high-PHY of a 5G NR protocol stack. Specifically, we design the system to harness GPU resources available as part of the NVIDIA Aerial GPU-accelerated platform. Aerial already uses GPU to offload computationally intensive and time-sensitive high-PHY layer workloads, contributing approximately 85% of the overall computational complexity and processing demands in gNBs [14]. This architecture enables efficient resource sharing, allowing the dApp to perform inference with minimal computational overhead while preserving real-time processing capabilities. As explained in Sec. 4, InterfO-RAN is implemented through customized functions utilizing a CNN architecture, executed on a GPU for real-time interference detection. The output of InterfO-RAN is a binary indicator that denotes the presence of interference. Once interference is detected, different strategies can be put in place for mitigation (e.g., coordinated resource allocation), which are left for future work.

The system processes input features comprising a combination of raw I/Q samples within a slot carrying PUSCH data, along with RSSI, RSRP, the number of Code Block (CB) errors, the total count of CBs, SINR, Modulation and Coding Scheme (MCS) index and MCS table index. The I/Q samples are extracted from the PUSCH physical channel, without any processing such as equalization, demodulation, or decoding. InterfO-RAN does not interfere with ongoing UL/DL processes, let alone PUSCH channel processes, while determining whether transmissions are affected by unwanted signals within the same frequency band.

Figure 2 further illustrates the UL interaction between gNB and UE, providing a comprehensive view of the data flow and the PUSCH channel engaged by InterfO-RAN. The PUSCH carries data, control information, and DMRS for channel estimation. The figure also highlights the other uplink channels, i.e., Physical Uplink Control Channel (PUCCH), Sounding Reference Signal (SRS), and Physical Random Access Channel (PRACH).

In the next sections, we provide a preliminary overview of the GPU-accelerated framework for physical layer processing, and then detail how InterfO-RAN is designed and implemented in detail.

3 GPU-Accelerated Physical Layer Processing

This section provides a detailed overview of the framework we leverage as the basis for the InterfO-RAN implementation, as well as notions on the physical layer PUSCH.

3.1 NVIDIA Aerial Physical Layer

InterfO-RAN is embedded as a functional plugin in NVIDIA Aerial CUDA Baseband (cuBB), a Software Development Kit (SDK) developed by NVIDIA that provides a 5G signal processing pipeline for GPUs, implemented in CUDA and C++ [15]. cuBB operates with slot-level granularity, with each slot lasting 500 μs, aligned with a 30 kHz subcarrier spacing. InterfO-RAN complements cuBB with real-time, high-performance inference. NVIDIA Aerial uses a GPU for inline acceleration of resource-intensive tasks such as channel estimation, Low Density Parity-Check Code (LDPC) encoding/decoding, and channel equalization, among others.

From an implementation standpoint, cuBB combines several software components, which we extend to design and implement InterfO-RAN. The CUDA Physical layer (cuPHY) controller serves as the primary coordinator, initializing GPU resources and creating the initial context for RU connections. During OTA UL/DL transmissions, the L2 adapter within the cuPHY controller translates control plane messages from the Small Cell Forum (SCF) Functional Application Platform Interface (FAPI) interface into slot commands for PHY-layer data traffic. These commands are processed by the cuPHY controller, converted into tasks, and assigned to specific UL and DL channel worker threads. Mapped to appropriate Central Processing Unit (CPU) cores, these threads delegate computational tasks to the GPU. Each thread represents a physical channel implemented as a pipeline, with operations executed on CUDA kernels. CUDA kernels are functions executed in parallel on a GPU. The pipeline is supported by Application Programming Interfaces (APIs) provided by the cuPHY library, invoked by the cuPHY driver to manage creation, configuration, and execution.

3.2 PUSCH Operations

InterfO-RAN focuses on the UL receive path, particularly the PUSCH, to detect interference. UL processing starts with slot configuration from the L2 layer and ends with UL signal processing. Among UL channels (PUSCH, PUCCH, PRACH, SRS), InterfO-RAN analyzes the PUSCH, the primary UL channel for data transmission, to detect interference from other UEs sharing the same radio resources.

PUSCH transmissions are dynamically scheduled by the gNB, which allocates frequency and time resources based on real-time network conditions. The resource allocation is signaled to the UE through Downlink Control Information (DCI) messages, specifying parameters such as MCS, resource block allocation, and transmission power [16]. The PUSCH channel in cuBB is implemented as a pipeline, for operations such as Resource Element (RE) demapping, channel estimation, channel equalization, de-rate matching, de-layer mapping, de-scrambling, LDPC decoding, and Cyclic Redundancy Check (CRC) checks for both CBs and Transport Blocks (TBs). Additionally, it includes processing for UCI on PUSCH and transform precoding, if enabled, executed through individual CUDA kernels. Specialized kernels also manage Timing Advance (TA) and Carrier Frequency Offset (CFO).

This pipeline is further optimized and structured using an advanced feature known as CUDA graph, which represents an acyclic graph where nodes correspond to kernels, and edges define their inter-dependencies, as illustrated in Fig. 3. This graph-based architecture effectively delineates the intricate workflow of kernel operations, enabling the management of dependencies within the graph itself. By facilitating the concurrent launch of all kernels as a unified entity, the GPU autonomously oversees the execution process, thereby reducing the necessity for continuous CPU intervention. This method significantly enhances temporal efficiency by minimizing launch overhead, which is the latency associated with initiating CUDA kernels.

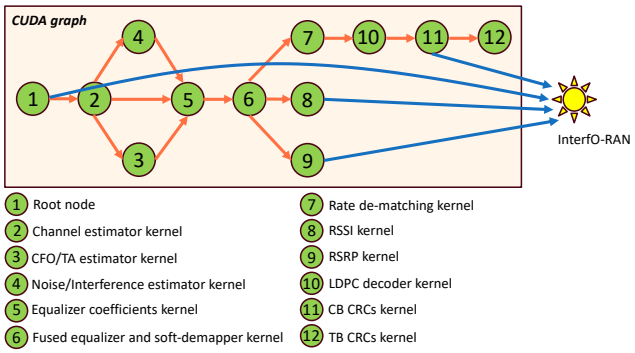


Figure 3: InterfO-RAN as a post-processing component following the PUSCH CUDA graph, where each node represents a CUDA kernel.

4 InterfO-RAN Design and Implementation

We design and implement InterfO-RAN on top of the NVIDIA Aerial SDK, leveraging the spare capacity of the GPU to execute the CNN-based dApp. As discussed in detail in the results analysis in Sec. 6, the typical execution of the PHY layer in cuBB utilizes at most 50% of the GPU resources, leaving substantial computational capacity available for additional tasks. This excess capacity enables InterfO-RAN to take advantage of the remaining GPU resources for interference detection. In doing this, however, it becomes necessary to design the dApp to efficiently access the available resources and to avoid interfering with real-time processing constraints of the base station physical layer.

4.1 dApp Design and PHY Integration

Design and Implementation Challenges. To design and integrate InterfO-RAN within the 5G NR GPU-accelerated PHY, we addressed the following challenges. First, the dApp needs to access information available across different processing steps of the PUSCH pipeline (i.e., I/Q samples, RSRP, RSSI, the sum of CB CRC errors, count of CBs derived from CB CRCs, SINR, MCS Index and MCS table index). This can amount to 183,484 bytes for each UL transport block, thus requiring an efficient mechanism to expose such information. Second, the CNN needs to return results in less than a millisecond (i.e., a 5G NR subframe) to make sure that the output is relevant to take further decisions across the stack. Therefore, the GPU-based implementation needs to be efficient in running inference with the provided input. In addition, there may be a need to support different models or configurations for the AI processing, thus managing the lifecycle of the overall model. For these reasons, we select ORT as the provider, making it possible to efficiently deploy trained models on the GPU-based dApp. At the same time, ORT requires a significant loading and configuration time the first time it is executed. In general, as part of our design, it is important to avoid disrupting operations of the rest of the physical layer, thus guaranteeing that the timing of any operation within the dApp does not affect physical layer processing. Finally, for development purposes, the InterfO-RAN implementation needs to enable automated data collection and labeling, to streamline the gathering of samples for the training of AI models.

InterfO-RAN System Structure. As the core component of InterfO-RAN, the inference module functions as a post-processing unit following the execution of the PUSCH pipeline, ensuring that ongoing UL/DL pipeline processes remain unaffected, as illustrated in Fig. 3. This functionality is efficiently implemented through the function `CuPhyInferCuDnn()`—a customized API hosted by cuPHY and managed by the cuPHY driver, as indicated in Fig. 4. This API further employs the `Session.Run()` method from the ORT framework to execute the actual inference process. To prevent InterfO-RAN from interfering with the tightly time-coupled operations of the PUSCH or other channels, the cuPHY driver delegates the processing to a newly instantiated and independent CPU thread. This thread is assigned to a dedicated CPU core—isolated from other operations—to improve stability.

Table 1: Inference time between different execution providers for standalone ORT.

ORT Execution Provider	CPU	CUDA	TensorRT
Inference Time (ms)	30.354	3.675	0.524

ORT Integration and Tuning. ORT is built to execute models in the Open Neural Network Exchange (ONNX) format, which is an open standard for representing ML models. AI operations using ORT can be executed on either the CPU, utilizing the CPU execution provider, or on the GPU, utilizing CUDA or Tensor RealTime (TensorRT) execution providers. Our implementation supports all three options, with processing latency shown in Tab. 1, for inference using standalone ORT (i.e., independently of the dApp implementation) and the model discussed in Sec. 4.2. CPU-based inference takes approximately 30.354 ms for a 24-core Intel Xeon Gold CPU, while GPU-based inference using TensorRT and CUDA takes 0.524 ms and 3.675 ms using an NVIDIA A100, respectively.

Although both CUDA and TensorRT utilize the GPU for inference tasks, TensorRT is specifically optimized for high performance. NVIDIA TensorRT is an SDK for deep learning model optimization, featuring an inference optimization engine and runtime environment. As evidenced in Tab. 1, it achieves lower execution times than both CPU and CUDA providers. The framework parses the ONNX model, applies optimizations such as layer fusion, and selects efficient CUDA kernels by leveraging libraries like CUDA Deep Neural Network library (cuDNN) and CUDA Basic Linear Algebra Subroutines (cuBLAS). Consequently, TensorRT is used exclusively in our experiments to maximize performance.

Based on our measurements, however, the GPU-based ORT mode comes with a significant delay during initial execution—1.25 s and 9.65 s with CUDA and TensorRT, respectively—for the model described in Sec. 4.2. These findings identify the root cause of the timing issue as the long GPU warm-up phase. This is because ORT requires several seconds to initialize and allocate GPU resources essential for inference. In particular, during the first call to `Session.Run()`, ORT performs several critical setup steps: (i) Memory allocation for the model tensors; (ii) Graph optimization and compilation to ensure efficient runtime execution; (iii) Execution provider setup, such as configuring CUDA for GPU inference; and (iv) Caching mechanisms for future executions. This warm-up latency conflicts with the strict timing constraints of the UL slot in 5G NR systems, making it a challenge for real-time execution. However, once initialized, these setup steps are cached, allowing subsequent invocations to bypass the preparatory steps and run significantly faster.

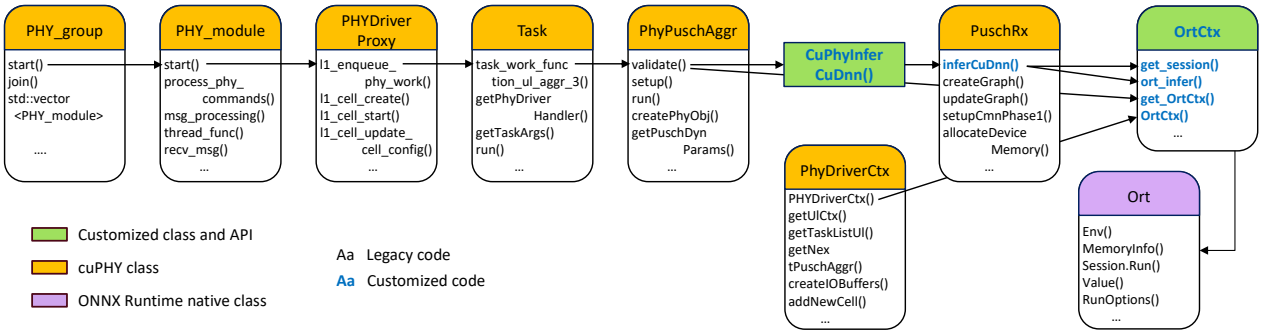


Figure 4: The workflow diagram for the OrtCtx class for performing inference.

PHY Layer and dApp Setup. Therefore, we design the InterfO-RAN dApp so that it performs an initial warm-up inference before the base station becomes operational. Particularly, during the initialization of cuBB, a preliminary inference step is performed on dummy data encompassing all input features. Then a pointer to the ORT run session (already initialized) is passed to InterfO-RAN. To support this, InterfO-RAN is encapsulated within a customized ORT context class, `OrtCtx`, which is instantiated within the cuPHY driver context class, `PhyDriverCtx`, as illustrated in Fig. 4. This manages instances of transmit and receive processing pipelines for UL and DL slots, as well as individual classes corresponding to each PHY channel, among others. Additionally, it handles the allocation of GPU resources and manages the GPU footprint for all UL and DL executions. `PhyDriverCtx` is instantiated as the whole physical layer instance starts, making it possible to execute the warm-up phase for ORT.

As part of this process, the `OrtCtx` class constructor initializes and configures the dynamically allocated instance of an ORT inference session as a smart pointer, using the native ORT API and leveraging its comprehensive classes and methods, as shown in Fig. 4. The process starts with creating an `Ort::Env()` object for logging and runtime management, followed by configuring session options for optimization, including graph processing, threading, memory allocation, and GPU-specific settings for CUDA and TensorRT. The dummy input data for warm-up inference is allocated in CPU memory and transferred to the GPU for execution, encapsulated as `Ort::Value()` tensors. `Session.Run()` processes the data through the pre-trained ONNX model and generates output tensors containing class probabilities, selecting the highest probability as the final prediction.

Runtime Inference. Upon initialization, the `OrtCtx` object is passed to the PUSCH channel via the physical PUSCH aggregate class, `PhyPuschAggr`, and invoked during the validation phase as a callable function, `get_OrtCtx()`. This validation phase is a post-processing step always executed after the PUSCH pipeline. Inference in the PUSCH pipeline is orchestrated by `inferCuDnn()`, a member of `PuschRx` class, and invoked by `CuPhyInferCuDnn()` API, as illustrated in Fig. 4. It retrieves the inference session from the `OrtCtx` object via `get_session()`, reusing the session configuration and runtime environment. We encapsulate input features into ONNX-compatible tensors using `ort_infer()`, using GPU memory for CUDA/TensorRT or CPU memory otherwise. When performing inference on the GPU, the input features are transferred directly within GPU memory without overhead. Tensors follow predefined shapes and types, with input/output names built dynamically for runtime flexibility.

dApp-based Automated Data Collection. We also designed the InterfO-RAN dApp to perform automated data collection to create datasets for offline model training. This leverages a `CuPhyInferCuDnn()` API to stream data into an Hierarchical Data Format version 5 (HDF5) file at a frequency of one every 10 UL slots. Different from what happens for online inference, the API asynchronously transfers features to be stored from GPU to CPU and writes to disk thereafter. This avoids many GPU operations, including data conversion to tensors and subsequent GPU memory deallocation. Besides, to avoid interference with the tightly coupled operations of PUSCH or other channels, the cuPHY driver delegates data logging to a newly instantiated CPU thread operating in detached mode, ensuring uninterrupted processing.

4.2 AI-Based Interference Detection

InterfO-RAN leverages a CNN for real-time interference classification, using a data-driven methodology that allows the network to identify complex patterns in incoming UL data.

Input Features. As discussed above, the input features include I/Q samples for the PUSCH, as well as additional Key Performance Measurements (KPMs) representing RSSI, RSRP, SINR, MCS index, MCS table index, total count of CB errors, and number of CBs derived from CB-CRCs. As part of the dApp design, one challenge was ensuring the compatibility of heterogeneous numerical formats, such as *float16* (for I/Q), *float32* (for RSSI, RSRP, SINR), and *int32* (for MCS and CB inputs), across frameworks such as Tensorflow (TF) (offline training) and cuBB.

Unlike traditional approaches that rely on heatmaps of I/Q samples for interference detection [17, 18], InterfO-RAN processes raw I/Q data directly and incorporates additional features to improve classification accuracy. I/Q samples are extracted prior to the Minimum Mean Square Error (MMSE)-Interference Rejection Combining (IRC) equalizer, which is designed to mitigate channel distortion and interference. This pre-equalization extraction provides an unaltered view of interference effects, allowing for more accurate analysis. In cuBB, the PUSCH channel I/Q samples are stored as 32-bit words (with two 16 bits floats for each component), in blocks of $14 \times 273 \times 12$ contiguous words, where 14 is the number of symbols per slot, 273 is the maximum number of Physical Resource Blocks (PRBs) with 12 subcarriers each. ORT converts the I/Q samples into a matrix with dimensions 14, 3276, 2, to align with

TF's expected format for inference, interpreting the third dimension as the real and imaginary components of the I/Qs.

Additionally, specific input features are transformed to ensure data compatibility. In the native CB-CRC CUDA kernel, the CB errors for each PUSCH channel are output as an array, where each non-zero value represents a corrupted CB within the TB, resulting in a dynamic size. To meet the fixed-size input requirements of the learning model, two additional variables are added to the GPU: one to determine the number of corrupted CBs in a TB and another to find the total number of CBs. These operations are seamlessly integrated into the kernel 11 in Fig. 3.

CNN Architecture. We choose CNNs for their ability to extract complex patterns from OFDM-based IQ samples and their consistent outperformance of alternatives like LSTMs and ResNets. Fig. 5 illustrates the selected architecture. The network features seven layers. Two convolutional blocks perform feature extraction on PUSCH I/Q samples, each containing two Conv2D layers (128 and 256 filters, 3×3 kernel, ReLU activation) and one 2×2 MaxPooling layer. Then, the extracted features are flattened and concatenated with the additional normalized structured inputs (i.e., RSSI, RSRP, SINR, MCS index, MCS table index, total count of CB errors, and number of CBs). This final representation is processed through a fully connected Dense layer with softmax activation, which performs the final interference detection. The starting point of our proposed CNN is the VGG16 model, a widely recognized architecture in image processing. We systematically reduced its depth and identified a two-block variant that provided the best trade-off between computational efficiency and classification performance.

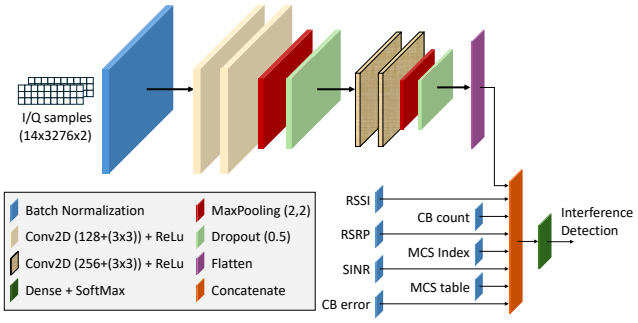


Figure 5: InterfO-RAN's CNN architecture showing inputs (I/Q and scalar features), layer specifications, and the binary interference detection output.

To enhance generalization and mitigate overfitting, we integrate dropout layers and an L2 regularizer. Dropout layers randomly deactivate neurons during training, encouraging the model to learn underlying patterns rather than memorizing the training data, while the L2 regularizer penalizes large weights, promoting simpler, more generalized models. Additionally, to address class imbalances (e.g., in case of data collections with different number of samples for different radios, or interference conditions), we use a weighted loss function instead of a standard one, which tends to favor the majority class. By assigning a greater weight to the minority class, the loss function ensures that its contributions have a stronger influence on the overall loss computation.

Transfer Learning. We employ TL using heterogeneous datasets from diverse Radio Frequency (RF) environments, detailed in Sec. 5.2, to enhance model generalization capabilities. TL is a machine learning technique wherein knowledge acquired from a source domain is leveraged to improve performance on a target domain through model adaptation. In our implementation, we partially fine-tune the foundational CNN model (Fig. 5), trained on data from multiple deployment location (see Fig. 6), and then tune to specific deployments by freezing the first block and fine-tuning the remaining layers with site-specific data. Thus, we allow the model to learn from a broader range of conditions and fine-tune it for a specific deployment, retaining knowledge of diverse scenarios by preserving foundational feature representations and enhancing performance.

5 Data and Evaluation Framework

This section presents the setup for the training, testing, and evaluation of InterfO-RAN, from the generation of synthetic data with simulations to experimental OTA data collection and evaluation. We also discuss preprocessing and offline training procedures.

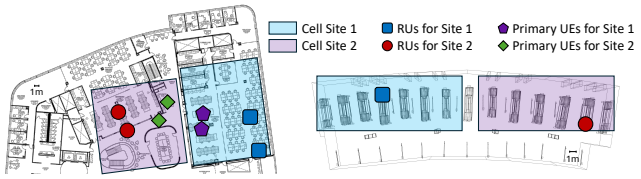
5.1 Primer on X5G

Real-world data collection, experiments, and validation of InterfO-RAN are performed using the X5G platform. X5G is an O-RAN-compliant, multi-vendor private 5G network featuring a multi-node deployment of the NVIDIA Aerial framework [15], spanning the entire floor of the EXP building on the Boston, MA, campus of Northeastern University [12]. The platform is fully open and programmable, allowing researchers to modify any part of it for experimentation and testing, including the integration and development of dApps like InterfO-RAN, within a testbed offering production-ready performance and capabilities. Each node features OAI for the upper layers of the protocol stack (CU and DU-High) and the NVIDIA Aerial SDK for the DU-Low. A structured networking infrastructure, consisting of various switches and connections, enables switching between RUs from different vendors, including Foxconn operating in the n78 band. Additionally, the Core Network (CN) is sourced from various open projects, such as Open5GS. X5G supports a range of Commercial Off-the-Shelf (COTS) UEs, including OnePlus, iPhone, and Samsung, as well as 5G modules like Sierra boards. Finally, it integrates the O-RAN Software Community (OSC) near-RT-RIC (RIC) for xApp development. All X5G RAN software runs inside Docker containers on dedicated RAN servers, including Gigabyte E251-U70 machines. Each Gigabyte server is equipped with a 24-core Intel Xeon Gold CPU, 96 GB of RAM, and a Broadcom Peripheral Component Interconnect (PCI) switch with two slots, hosting an NVIDIA A100 GPU and a Mellanox ConnectX-6 Dx Network Interface Cards (NICs). This setup enables direct connectivity, bypassing CPU involvement, while ensuring high transfer speeds and providing the necessary GPU computational power to run the RAN software along with additional workloads, such as InterfO-RAN.

5.2 Empirical OTA Data Collection, Automated Labeling, and Preprocessing

All experiments involving interference are conducted in a controlled indoor environment within the Northeastern University EXP building in Boston, MA, as depicted in the indoor experiment map shown in Fig. 6a. The layout includes: two cell sites (shown in light blue and pink regions), where the corresponding UEs can be located; the

RU locations for each cell site, with only one active at a time per site (represented by blue and red icons); and the two primary UE locations for the two cell sites, where most of the data is collected (violet and green icons). Each experiment involves both cell sites and one UE per cell at a time, tested under Line-of-Sight (LOS) and NLOS channel conditions. Similarly, we collected additional data in a second indoor environment within the Northeastern University ISEC building in Boston, MA, (Fig. 6b) as mentioned in Sec. 4.2.



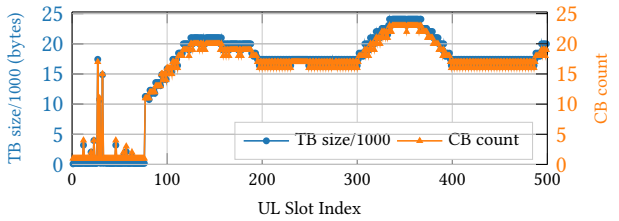
(a) EXP Environment (b) ISEC Environment

Figure 6: Indoor experiment layout maps of Northeastern University EXP (a) and ISEC (b) buildings, showing: the designated regions for the two cell sites, where the corresponding UEs can be located; RU locations for each respective cell site; and primary UE locations, where most of the data is collected.

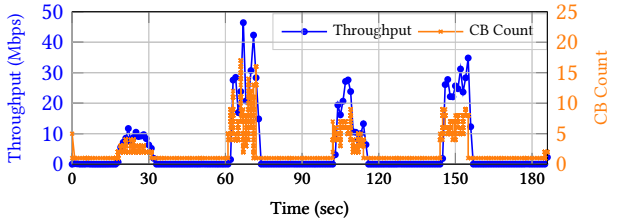
During each data collection session, the UE connects to its RU and generates UL traffic using iPerf. We categorize the UL traffic levels at the gNBs into two distinct classes—High Traffic and No Traffic—based on the TB size, to improve interference identification accuracy. In the No Traffic scenario, the UE remains connected to the gNB while exchanging minimal control information necessary for maintaining the connection stability. In this case, we observe that the TB size is typically around 185 bytes and does not exceed 1056 bytes, with the number of CBs limited to one. The CBs from a TB are segmented and LDPC-encoded using base graphs defined by the 3GPP for error correction. LDPC base graph 2 is used for smaller CBs, with each CB sized at 480 bytes, while base graph 1 is used for larger ones, with each CB sized at 1056 bytes. In contrast, High Traffic UL transmissions produce sufficiently strong signals that can interfere with neighboring cells. In this scenario, the TB size typically exceeds 1056 bytes, and the number of CBs is greater than one.

As depicted in Fig. 7, there is a strong correlation between the TB size and the number of CBs (Fig. 7a), as well as between the UL throughput and the CB count (Fig. 7b), reinforcing the decision to use the CB count as the primary criterion for traffic classification. Building upon this, we create a systematic labeling methodology based on the UL traffic levels to classify the UL transmissions as either affected by interference ('INTERF') or unaffected ('CLEAN'). The labeling rules, shown in Tab. 2, are defined as follows:

- No Traffic/High Traffic: if one UE transmits with 'High Traffic' while the other has 'No Traffic', the 'High Traffic' time windows on one side are used to label the other side as being affected by interference, designated as 'INTERF', while the transmitting side is labeled as 'CLEAN'.
- High Traffic on both sides: if both UEs transmit simultaneously to different RUs with 'High Traffic', the corresponding samples are labeled as 'INTERF'.
- No Traffic on both sides: if both UEs are transmitting with 'No Traffic', the samples are labeled as 'CLEAN', i.e., no interference.



(a) TB size in bytes (scaled down by 1000) versus CB count



(b) UL throughput versus CB count

Figure 7: UL throughput, TB size, and CB count across different scenarios, demonstrating their close correlation.

Table 2: Lookup Table for gNB UL Traffic and Labels

gNB1 Traffic	gNB2 Traffic	gNB1 UL Label	gNB2 UL Label
High Traffic	No Traffic	CLEAN	INTERF
No Traffic	High Traffic	INTERF	CLEAN
High Traffic	High Traffic	INTERF	INTERF
No Traffic	No Traffic	CLEAN	CLEAN
No UE	No/High Traffic	NA	CLEAN
No/High Traffic	No UE	CLEAN	NA

- Single-active Transmission: if only one UE transmits while the other remains in airplane mode (i.e., not connected to the base station), the samples are labeled as 'CLEAN'.

A threshold of 1 is set for the CB count in an UL slot to identify high-traffic windows on one gNB, which are subsequently used to mark instances of interference on the other gNB. Notably, the exact thresholding mechanism is utilized to evaluate inference performance in OTA data scenarios by identifying high-traffic time windows, which are then used to assign ground truth labels of 'CLEAN' or 'INTERF' to each input example, thereby enabling the effective monitoring of spectrum sensing capabilities.

To efficiently manage GPU memory during offline training—particularly given the relatively large input feature consisting of 3D I/Q samples of size $14 \times 3276 \times 2$—we use TF's Sequence class to implement a custom data generator. The data generator dynamically loads small batches into GPU memory, rather than loading the entire dataset at once, thereby optimizing memory usage and avoiding excessive memory strain.

5.3 Synthetic Data

We also perform an extensive data collection campaign using the NVIDIA Aerial simulator nr_sim, which complements the cuBB framework by extending the 5G MATLAB Toolbox. Through this, we collect data and evaluate the model in a larger variety of configurations compared to those supported in our experimental setup. The parameters that we sweep include Signal-to-Noise-Ratio (SNR), Signal to Interference Ratio (SIR), MCS index, channel type, delay profile, interference channel delay profile, numerology, and carrier frequency. In addition, we simulate up to 5 interfering UEs, so that the classification can be extended to include additional UEs.

To simulate interference, an in-band interference signal is artificially generated by transmitting Orthogonal Frequency Division Multiplexing (OFDM) signals from each antenna independently through a Tapped Delay Line (TDL) channel characterized by high delay spread and Doppler shift. These conditions emulate a dynamic and realistic environment affected by multipath propagation and high mobility, thus creating a robust testbed for interference mitigation. The processed signal is then multiplied by a modified all-zero matrix with randomly distributed contiguous ones, introducing localized and burst interference in the REs of the OFDM slot. This process ensures random patterns for contiguous ones across antenna streams while maintaining the average power of the interference signal. The resulting combination of randomized interfering signals, thermal noise, and received OFDM signal at the gNB closely mimics real-world interference effects.

6 Performance Evaluation and Experimental Results

This section presents experimental results to benchmark the performance of InterfO-RAN across various model configurations, using the simulation setup described in Sec. 5.3 as well as over OTA tests.

We test the model, shown in Fig. 5, on synthetic data with scenarios involving 0 to 5 interferers, totaling 13800 samples per class, to ensure consistent performance across different parameters described in Sec. 5.3 before OTA deployment in InterfO-RAN. The resulting confusion matrix in Fig. 8 indicates that the model achieves high accuracy (above 96%) for scenarios with no interferer and 2 or more interferers. For scenarios with a single interferer, accuracy remains solid at 93.87%, with the slight drop likely due to the lower impact of a single interferer on the signal. Thus, the model becomes a strong candidate for OTA deployment, along with several other models with minor modifications explained in Sec 6.1.

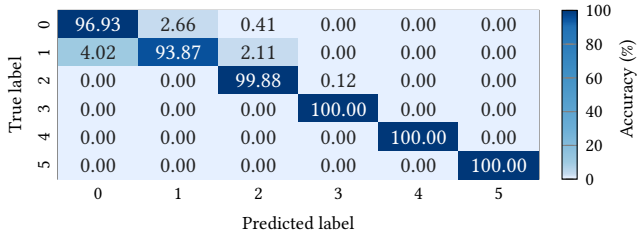


Figure 8: Confusion matrix of the model tested on synthetic data with 13800 samples per class.

6.1 OTA Evaluation

To select the best generalized model suitable for diverse RF environments, we conduct a series of OTA experiments evaluating eight configurations of the underlying CNN architecture shown in Fig. 5. We represent each model as $\{[\alpha,\beta],\gamma\}$, where α and β denote the number of filters in the convolutional layers of the first and second blocks, and γ represents the batch size. We select two configurations for $[\alpha, \beta]$: [64, 128] and [128, 256], inspired by VGG16's first blocks for a lightweight model, with γ ranging over 16, 32, 64, and 128.

We carry out experiments at various RU locations and UE positions across the map, as shown in Fig. 6. For fair comparability, we evaluate different models using the same dataset with data logging. The data collected from one side is labeled with ground truth based on the traffic level on the other side, as detailed in Tab. 2. We then compute metrics such as accuracy, measuring the ratio of correctly predicted samples to the total number of samples, specificity, evaluating the identification of 'CLEAN' examples, and recall, assessing the detection of 'INTERF' examples.

(a) Performance in a familiar RF environment

(b) Performance in an unseen RF environment

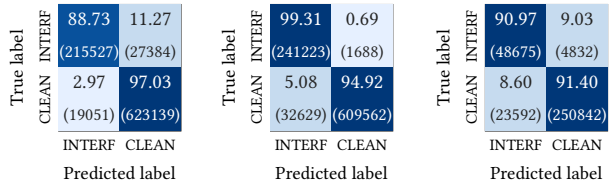
Figure 9: Model performance (accuracy, specificity, and recall) for Transfer Learning: (a) familiar RF environment and (b) unseen RF environment to evaluate generalization.

Fig. 9 compares accuracy, specificity, and recall for TL models, described in Sec. 6.2, tested in a familiar RF environment (Fig. 9a), and an unseen RF environment (Fig. 9b), to evaluate generalization capabilities. We select the {[128,256], 32} model as the best-performing configuration, based on its strong results in familiar settings (96.12% accuracy, 94.92% specificity, and 99.31% recall) and its superior generalization in the unseen RF environment (91.33% accuracy, 90.97% specificity, and 91.40% recall), outperforming the other model configurations.

6.2 Impact of Transfer Learning

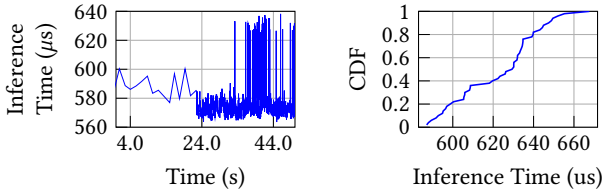
We evaluate InterfO-RAN performance across various RF configurations to assess the impact of the TL method on the model. We use ISEC dataset (Fig. 6b), dominated by High Traffic instances on both sides, as the foundational model for TL, then fine-tuned on the EXP dataset (Fig. 6a), detailed in Sec. 4.2, which contains sufficient data for all traffic instances, as described in Tab. 2. Additionally, we train a baseline model using only the EXP dataset, and compare both performances to assess the benefits of TL.

Fig. 10a shows the confusion matrix of the baseline model without TL, tested in a familiar RF environment using the same pair of RUs locations as in the training. In contrast, Fig. 10b and Fig. 10c present the performance of the TL model, evaluated both in the same familiar RF setting as the non-TL model, and in a new set of RU locations, i.e., an unseen environment. The TL model achieves overall better performance in detecting interference when tested in both a familiar (+10.52%) and unseen (+2.24%) RF. However, it is outperformed by the non-TL model in No Traffic scenarios, with



(a) No TL in familiar RF environment (b) TL in familiar RF environment (c) TL in unseen RF environment

Figure 10: Classification accuracy (percentage, sample count) with and without TL across different RF environments: (a) no TL in a familiar environment, (b) TL tested in a familiar environment, and (c) TL tested in an unseen environment, using the same color bar as Fig.8



(a) Average inference time (b) Inference time CDF

Figure 11: Inference time from first iteration: (a) temporal variation with a 5-instance moving average with No (before 24 s) and High Traffic (after 24 s), (b) inference time CDF.

Table 3: Average inference times across different models.

Model Configuration	{[64,128],any}	{[128,256],any}
Inference Time (μs)	401.8	621.6

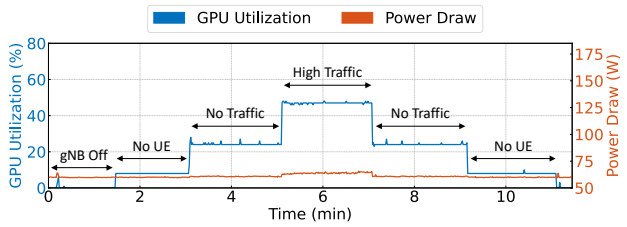
performance drops of -2.11% in the familiar and -5.63% in the unseen one, likely due to the influence of the foundational model.

6.3 Timing and Power Benchmarking

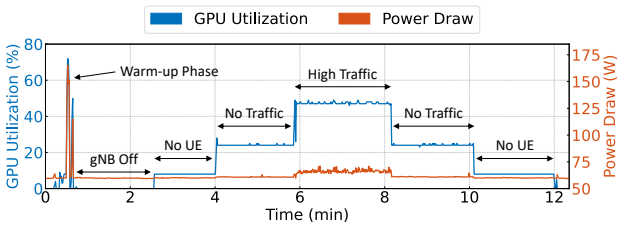
We assess InterfO-RAN's robustness by analyzing its inference time, GPU utilization, and power consumption, both with and without it.

Analysis of Inference Time. Fig. 11a shows the temporal variation of the inference time using a 5-instance moving average between No (before 24 s) and High Traffic (after 24 s), while Fig. 11b presents the corresponding Cumulative Distribution Function (CDF). We observe fluctuations during high-traffic scenarios, when InterfO-RAN performs more frequent inferences, likely due to the increased number of operations and contention for GPU resources. However, the system maintains overall stability, effectively managing workload distribution. Additionally, Tab. 3 shows the average inference time for different model configurations, varying with filter counts. We notice a 220 μs improvement in the smaller configuration due to the reduced computational complexity.

Analysis of GPU Utilization and Power. Fig. 12a and Fig. 12b compare A100 GPU utilization and power draw with and without InterfO-RAN in the same end-to-end scenario. During the warm-up phase (see Sec. 4.1), GPU usage spikes to 21.2% (vs. 3.3% without InterfO-RAN), and the power rises to 82.73 W (vs. 61.79 W). In other phases, power remains similar, except in High Traffic conditions, where it increases from 63.83 W to 66.13 W and a higher standard deviation (1.77 vs. 0.67) when using InterfO-RAN. We conclude that InterfO-RAN effectively balances the workload without straining RAN operations, while leveraging TensorRT optimizations.



(a) Without InterfO-RAN dApp



(b) With InterfO-RAN dApp

Figure 12: GPU utilization (blue) and power draw (orange) comparison with and without InterfO-RAN dApp.

7 Related Work

Limited research has been conducted on UL interference in 5G NR within real-time timescales and tested on real-world testbed environments. Existing literature predominantly relies on simulations to predict the presence of interference. For instance, [19] and [20] detect intermodulation interference in 5G NR using linear regression and CNNs, respectively. Similarly, [17] and [18] detect Long Term Evolution (LTE) UL interference using a novel approach that preprocesses time-domain signals into spectral waterfall representations and addresses the problem using an image classification CNN. Additionally, authors in [21] identify interference among IEEE 802.11b/g, IEEE 802.15.4, and IEEE 802.15.1 using a deep CNN.

In the context of spectrum monitoring, [22] develops a CNN for wireless signal identification, while [23] applies logistic regression to detect co-channel interference between LTE and WiFi users, though both approaches still lack validation beyond simulations. The authors of [24] propose a framework based on CNNs to sense and classify wideband spectrum portions, supported by real-world data collection and testing. Several studies are conducted to characterize interference signals in various scenarios, such as [25], which addresses downlink interference in massive Multiple Input, Multiple Output (MIMO) 5G macro-cells using geometric channel models, and [26], which focuses on modeling interference at higher frequencies for Sixth generation (6G) networks. Furthermore, [27] and [28] discuss interference mitigation mechanisms. The former focuses on angular-based exclusion zones and spatial power control in mmWave frequency ranges, while the latter employs supervised learning-based Interference Whitening (IW) selection methods.

Studies on timescales faster than those provided by non-Real-Time (over 1 sec) and near-Real-Time (between 10 and 1000 ms) RICs are continuously increasing, highlighting the need for faster and more adaptable control mechanisms capable of responding to dynamic network conditions. InterfO-RAN is based on the O-RAN concept of dApp operating on a timescale faster than 10 ms, first described in detail in [29]. This concept was recently expanded in [10], which presents a comprehensive framework built on the

OAI open-source project and further demonstrates its effectiveness through two use case applications, spectrum sharing and 5G positioning. Moreover, the authors in [30] extend their previous Open AI Cellular framework to support dApps, validating their approach through experimentation with a federated learning dApp. Real-time RAN control is enabled by [31] via a Real-Time RIC and μApp for intelligent MAC-layer scheduling at the DU. [32] integrates AI-driven decision-making, leveraging RAN and application data for sub-millisecond latency. CloudRIC [33] optimizes RAN scheduling within O-RAN using heterogeneous computing. However, to the best of our knowledge, InterfO-RAN represents the first end-to-end dApp implementation deployed on a GPU-based RAN node and tested in real-time on a production-ready, real-world testbed.

8 Conclusions and Future Work

This paper presents InterfO-RAN, a GPU-accelerated O-RAN dApp that detects in-band UL interference with over 91% accuracy at sub-millisecond speeds (650 μs). Our solution leverages a CNN to analyze I/Q samples directly within the gNB physical layer. InterfO-RAN is fully implemented in a commercial-grade private 5G network, featuring seamless integration with both NVIDIA Aerial's 5G NR stack and OAI. The system's effectiveness is validated through extensive evaluations on synthetic and real-world data, confirming its reliability and capabilities.

Future work on InterfO-RAN will address several key challenges. We will explore scalability to higher UE densities and mobility scenarios, repurpose it for AoA estimation/5G positioning, implement mitigation via resource allocation tuning to enhance performance, and develop cross-layer interfaces to enable seamless communication between the dApp and other network components. Finally, we aim to implement online training capabilities allowing the CNN to adapt continuously to specific RF environments and scenarios.

Acknowledgements

This work was partially supported by the U.S. NSF under awards CNS-2112471 and CNS-2434081 and by OUSD(R&E) through Army Research Laboratory Cooperative Agreement Number W911NF-24-2-0065. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-Dense Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2522–2545, 2016.

[2] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5G Ultra-Dense Cellular Networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[3] A. Narayanan, M. I. Rochman, A. Hassan, B. S. Firmansyah, V. Sathya, M. Ghosh, F. Qian, and Z.-L. Zhang, "A Comparative Measurement Study of Commercial 5G mmWave Deployments," in *IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[4] E. Eren, "Spectrum for private networks: Recent advancements by countries," 2025. [Online]. Available: https://tinyurl.com/as3cndjd

[5] F. B. Mismar, B. L. Evans, and A. Alkhateeb, "Deep Reinforcement Learning for 5G Networks: Joint Beamforming, Power Control, and Interference Coordination," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1581–1592, 2020.

[6] M. U. A. Siddiqui, F. Qamar, F. Ahmed, Q. N. Nguyen, and R. Hassan, "Interference Management in 5G and Beyond Network: Requirements, Challenges and Future Directions," *IEEE Access*, vol. 9, 2021.

[7] S. D'Oro, L. Bonati, F. Restuccia, and T. Melodia, "Coordinated 5G Network Slicing: How Constructive Interference Can Boost Network Throughput," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1881–1894, 2021.

[8] C.-Y. Chen, Y.-Y. Chen, and H.-Y. Wei, "Multi-cell interference coordinated scheduling in mmWave 5G cellular systems," in *Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016, pp. 912–917.

[9] W.-B. Yang, M. Souryal, and D. Griffith, "Lte uplink performance with interference from in-band device-to-device (d2d) communications," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, 2015, pp. 669–674.

[10] A. Lacava, L. Bonati, N. Mohamadi, R. Gangula, F. Kaltenberger, P. Johari, S. D'Oro, F. Cuomo, M. Polese, and T. Melodia, "dApps: Enabling Real-Time AI-Based Open RAN Control," *Computer Networks*, vol. 269, p. 111342, September 2025.

[11] AI-RAN Alliance, "AI-RAN Alliance: Vision and Mission White Paper," 2024. [Online]. Available: https://ai-ran.org/wp-content/uploads/2024/12/AI-RAN_Alliance_Whitepaper.pdf

[12] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R. S. da Silva, A. Kelkar, C. Dick, S. Basagni, J. M. Jornet, T. Melodia, M. Polese, and D. Koutsonikolas, "An Open, Programmable, Multi-vendor 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface," in *Proceedings of Next-generation Open and Programmable Radio Access Networks (NG-OPERA)*, Vancouver, BC, Canada, May 2024.

[13] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, Q2 2023.

[14] L. Kundu, X. Lin, E. Agostini, V. Ditya, and T. Martin, "Hardware Acceleration for Open Radio Access Networks: A Contemporary Overview," *IEEE Communications Magazine*, vol. 62, no. 9, pp. 160–167, 2024.

[15] NVIDIA, "Aerial RAN co-lab over-the-air (ARC-OTA)," 2025. [Online]. Available: https://docs.nvidia.com/aerial/aerial-ran-colab-ota/current/index.html

[16] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The Next Generation Wireless Access Technology*, 1st ed. Academic Press, 2018.

[17] J. Ren, X. Zhang, and Y. Xin, "Using Deep Convolutional Neural Network to Recognize LTE Uplink Interference," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[18] A. Medhat, M. Elattar, and O. M. Fahmy, "LTE Uplink Interference Inspection Using Convolutional Neural Networks," in *2021 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, 2021, pp. 318–322.

[19] F. B. Mismar, "Intermodulation Interference Detection in 6G Networks: A Machine Learning Approach," in *IEEE Vehicular Technology Conference: (VTC-Spring)*, 2022.

[20] F. Wu, R. Tan, C. Zhang, W. Fan, X. Chen, D. Niyato, and Y. Liu, "Mixed Numerology Interference Recognition Approach for 5G NR," *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2135–2139, 2021.

[21] M. Schmidt, D. Block, and U. Meier, "Wireless interference identification with convolutional neural networks," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 2017, pp. 180–185.

[22] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, 2018.

[23] L. Lai, D. Feng, and F.-C. Zheng, "Interference Detection and Resource Allocation in LTE Unlicensed Systems," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[24] D. Uvaydov, S. D'Oro, F. Restuccia, and T. Melodia, "DeepSense: Fast wideband spectrum sensing through real-time in-the-loop deep learning," in *IEEE Intl. Conf. on Computer Communications (INFOCOM)*, Vancouver, BC, Canada, May 2021.

[25] K. Bechta, C. Ziółkowski, J. M. Kelner, and L. Nowosielski, "Modeling of Downlink Interference in Massive MIMO 5G Macro-Cell," *Sensors*, vol. 21, no. 2, 2021.

[26] P. Testolina, M. Polese, J. Jornet, T. Melodia, and M. Zorzi, "Modeling Interference for the Coexistence of 6G and Passive Sensing Systems," *IEEE Transactions on Wireless Communications*, 2024.

[27] G. Hattab, E. Visotsky, M. C. Cudak, and A. Ghosh, "Uplink Interference Mitigation Techniques for Coexistence of 5G Millimeter Wave Users With Incumbents at 70 and 80 GHz," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, 2019.

[28] S. Chaudhari and H. Kwon, "Machine Learning based Interference Whitening in 5G NR MIMO Receiver," in *IEEE 95th Vehicular Technology Conference*, 2022.

[29] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dApps: Distributed Applications for Real-Time Inference and Control in O-RAN," *IEEE Communications Magazine*, vol. 60, no. 11, p. 52–58, Nov. 2022.

[30] M. Kouchaki, A. S. Abdalla, and V. Marojevic, "OpenAI dApp: An Open AI Platform for Distributed Federated Reinforcement Learning Apps in O-RAN," in *IEEE Future Networks World Forum (FNWF)*, Baltimore, MD, 2023.

[31] A. Bura, U. Ghosh, D. Bharadia, and S. Shakkottai, "Windex: Realtime Neural Whittle Indexing for Scalable Service Guarantees in NextG Cellular Networks," *arXiv:2406.01888 [eess.SY]*, 2024.

[32] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, and D. Bharadia, "EdgeRIC: Empowering Realtime Intelligent Optimization and Control in NextG Networks," *arXiv:2304.11199 [cs.NI]*, 2023.

[33] L. Lo Schiavo, G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024.