pyAerial

Table of contents

List of Figures

As 6G research gains momentum, and with many new technologies in its purvue, one thing is clear, AI/ML will feature prominently in the next generation RAN. It will play a pivotal role in realizing all parts of the network infrastructure from the radio units, baseband processing, the network core including system management, orchestration and dynamic optimization processes. GPU hardware, together with programming frameworks will be essential to realize this vision of a software defined native-AI communication infrastructure.

The application of AI/ML in the physical layer has in particular been a hot research topic. There is a lot of emphasis on neural network architectures and optimization strategies mostly performed in the context of simulation. The next step for the research community and commercial system developers is to bring AI/ML applied in layer-1 to reality in over-the-air real-time testbeds and operator-network scale systems.

This is where pyAerial enters the picture. pyAerial is a Python library of physical layer components that can be used as part of the workflow in taking a design from simulation to real-time operation. It helps with end-to-end verification of a neural network integration into a PHY pipeline and helps bridge the gap from the world of training and simulation in TensorFlow/PyTorch to real-time operation in an over-the-air testbed.

The pyAerial library provides a Python-callable bit-accurate GPU-accelerated library for all of the signal processing CUDA kernels in the NVIDIA cuBB layer-1 PDSCH and PUSCH pipelines. In other words, the pyAerial Python classes behave in a numerically identical manner to the kernels employed in cuBB because a pyAerial class employs the exact same CUDA code as the corresponding cuBB kernel: it is the CUDA kernel but with a Python API.

Using pyAerial library components complete layer-1 pipelines can be composed in Python. User code or inference engines, from NVIDIA TensorRT, or custom CUDA code, can be included in the datapath as shown in the lower part of Figure 1. This rapid prototyping design and verification flow is used for dataplane functional performance evaluation. It is a step in the workflow for verifying a physical layer design prior to deployment in a real-time over-the-air GPU base station.

pyAerial can also be used in conjunction with the NVIDIA data collection platform *Aerial Data Lake.* An Aerial Data Lake database consists of RF samples from a 7.2x fronthaul interface together with L2 meta-information to enable database search and query operations. A pyAerial pipeline can access samples from Aerial Data Lake database using the Data Lake Python APIs, and transform that data into training data for any function in the pipeline. Figure 2 illustrates data ingress from a Data Lake database into a pyAerial pipeline and using standard Python file I/O to generate training data for a soft de-mapper.

Content

Key Features

pyAerial has the following key features:

Feature 1: Productive Python for rapid prototyping of layer-1 pipelines

- pyAerial library components are CUDA kernels with Python bindings. The productive environment of Python permits the rapid assembly of signal processing pipelines in Python. All of the analytic and visualization aspects of Python can be used for performance characterization, signal visualization and debugging.

Feature 2: Simulate machine learning in the physical layer before over-the-air operation

- With the goal of going from model training and simulation in TensorFlow or PyTorch to real-time over-the-air operation, pyAerial provides a convenient way to verify, evaluate and benchmark your physical layer prior to deployment in an OTA testbed.

Feature 3: Fast simulation with CUDA optimized kernels

- pyAerial library components are CUDA under the hood. Simulation is fast on a GPU. When you are simulating the coding chain, including for example an LDPC decoder, optimized CUDA code is implementing these computationally heavy functions.

Feature 4: Generate data sets for any node in layer-1 uplink or downlink pipeline

- pyAerial is designed to be used in conjunction with the NVIDIA data collection platform *Aerial Data Lake*. pyAerial can access RF samples in a Data Lake database and transform those samples into training data for all of the signal processing functions in and uplink or downlink pipeline.

Feature 5: Bit accurate simulation

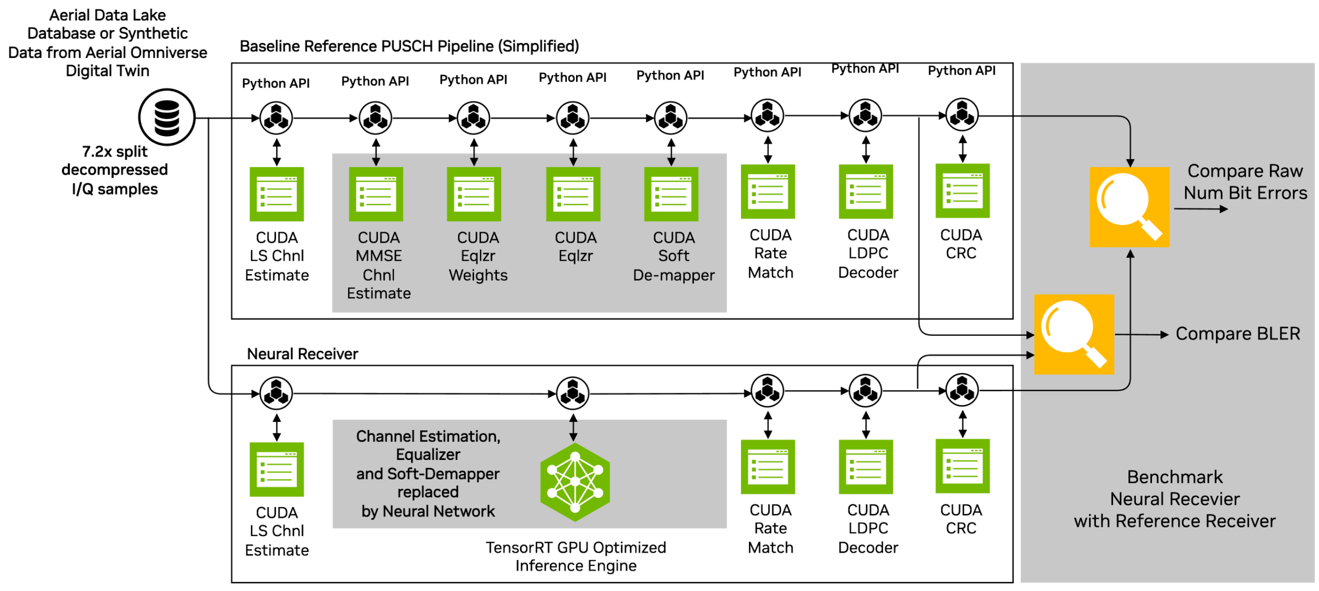
- Because pyAerial is Python running on CUDA, the performance you observe in BLER and other characterization metrics is what is identical to the performance of the real-time over-the-air system.

Target Audience

Industry and university researchers and developers looking to bring machine learning to the physical layer with the end goal of benchmarking on over-the-air testbeds like NVIDIA ARC-OTA or other GPU-based base stations.

Value Proposition

Fast bit-accurate GPU accelerated simulation of neural-network downlink and uplink signal processing pipelines. Rapid prototyping and functional verification of a real-time layer-1 in preparation for real-time deployment. Convenient Python environment aids debugging and provides easy access to all nodes in the pipeline for visualization and analysis. Easy to use Python environment for producing BLER and other statistics of interest for a real-time bit-accurate GPU layer-1 implementation. Transform RF sample captures for over-the-air captures into data for training layer-1 functions or compositions of multiple functions.



Aerial Data Lake Database or Synthetic Data from Aerial Omniverse Digital Twin
7.2x split decompressed I/Q samples
Baseline Reference PUSCH Pipeline (Simplified)
Python API
CUDA LS Chnl Estimate
CUDA MMSE Chnl Estimate
CUDA Eqlzr Weights
CUDA Eqlzr
CUDA Soft De-mapper
CUDA Rate Match
CUDA LDPC Decoder
CUDA CRC
Compare Raw Num Bit Errors
Compare BLER
Neural Receiver
Channel Estimation, Equalizer and Soft-Demapper replaced by Neural Network
TensorRT GPU Optimized Inference Engine
Benchmark Neural Recevier with Reference Receiver

Figure 1: Using pyAerial to verify a neural pipeline context of a full uplink pipeline. This is one of the verification steps to moving to real-time operation over-the-air on a GPU base station.

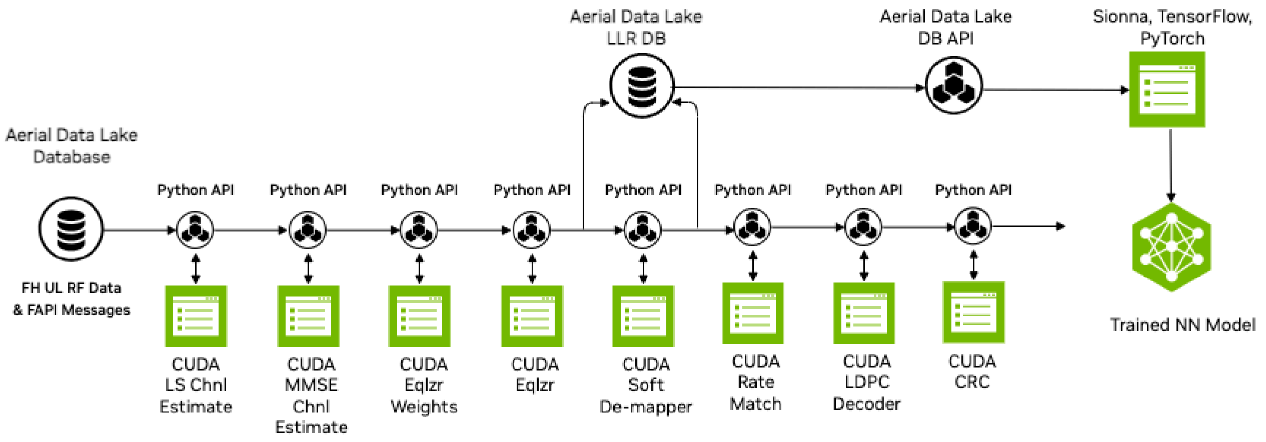


Figure 2: pyAerial is used in conjunction with the NVIDIA data collection platform Aerial Data Lake to build training data sets for any node in the layer-1 downlink or uplink signal processing pipeline. The example shows a Data Lake database of over-the-air samples transformed into training data for a neural network soft de-mapper, using pyAerial. Data gets extracted at the input and output of the de-mapper, and stored in the database.

Release Notes

- Release version: 24-1
- Supported configurations:
  - AX800, A100X and A100 GPUs with the x86 platform.
    - CUDA Toolkit: 12.2.0
    - GPU Driver (OpenRM): 535.54.03
  - Note: The Grace Hopper platform is currently not supported.
- Supported features: pyAerial exposes a subset of the cuPHY API features to Python. Currently this subset includes the following features:
  - PUSCH receiver pipeline
  - PDSCH transmission pipeline
  - Channel estimation
  - Noise and interference estimation
  - Channel equalization and soft demapping
  - LDPC encoding
  - LDPC decoding
  - LDPC rate matching

    - SRS channel estimation

- Limitations:
    - Unlike the cuPHY API, pyAerial API supports only a single UE group per method call. Multiple UE groups (FDM) can be supported by calling the methods separately for each UE group.