Real-time AI-enabled CSI Feedback Experimentation with Open RAN

Hai Cheng*, Pedram Johari*, Mohamed Amine Arfaoui†, Francois Periard†,
Philip Pietraski†, Guodong Zhang†, Tommaso Melodia*
*Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, U.S.A.
†Interdigital Inc., U.S.A.
*{cheng.hai, p.johari, t.melodia}@northeastern.edu
†{MohamedAmine.Arfaoui, Francois.Periard, Philip.Pietraski, Guodong.Zhang}@InterDigital.com

***Abstract*—There is interest from academia and industry to investigate the application of Artificial Intelligence (AI)/Machine Learning (ML) to various use cases associated with the Air Interface of cellular systems, e.g., for reporting Channel State Information (CSI) feedback, for beam management, and for positioning accuracy. In this paper, we develop a research platform capable of real-time inference using an AI-enabled CSI feedback that closely represents real-world deployment scenarios. In our experiment, we evaluate the performance of the proposed framework by integrating a CSI autoencoder into the OpenAir-Interface (OAI) 5G protocol stack. Further, we demonstrate the real-time functionality of the CSI compression framework with the encoder deployed at the User Equipment (UE) and CSI reconstruction with the decoder deployed at the Next Generation Node Base (gNB). The experiments are conducted on an Over-the-Air (OTA) indoor testbed platform, ARENA, as well as, on an emulated environment using Colosseum, the world's largest wireless network emulator.**

***Index Terms*—AI/ML, CSI feedback, Real-time experiments, Open Radio Access Network (Open RAN)**

I. Introduction

Inspired by the success of AI/ML applications in computer vision and natural language processing, research has moved towards evaluating the applications of AI/ML in wireless communication. The 3rd Generation Partnership Project (3GPP) Release 18 includes AI/ML for air interface as a Study Item [1]. The study aims to evaluate the potential benefit and associated complexity of a number of use cases, including AI/ML-based CSI compression and feedback. There have indeed recently been several works investigating the application of AI/ML into CSI compression [2]–[4]. However, all these works are based on simulation data, and to the best of our knowledge, no real-time experiments are conducted to compare efficiency and complexity based on results from practical systems.

In the past few years, applying ML techniques to address a variety of challenges in next-generation wireless networks has been widely discussed. Among those, re-designing the physical layer with AI-driven alternatives has drawn some attention [5]–[7]. Deep Learning (DL) was first introduced into the physical layer in [5]. A Neural Network (NN)-based communication system working on OTA transmission was developed in [6]. An NN-based WiFi Orthogonal Frequency Division Multiplexing (OFDM) receiver was proposed in [7]. An autoencoder-based CSINet was first proposed to compress CSI to reduce CSI feedback overhead in [2]. After that, various AI-based CSI feedback works were proposed [3], [4]. [8] provides a detailed review of AI/ML for CSI feedback for cellular systems.

In this paper, we develop a real-time AI-enabled CSI feedback experimental platform based on 5G Open RAN, and evaluate an AI-driven real-time CSI compression model as a practical use case. The developed platform is based on OAI open-source 5G protocol stack. An autoencoder-based CSI compression model, trained from MATLAB simulation data, is deployed based on Microsoft's Open Neural Network Exchange (ONNX) runtime. The model and runtime are both integrated into OAI-based platform to be evaluated on Arena, which is an indoor OTA testbed [9], as well as on Colosseum, which is the world's largest wireless network emulator with hardware-in-the-loop (HITL) [10]. We present the experimental results, including feedback overhead, CSI reconstruction accuracy, and inference latency, to evaluate the practicality and the performance of the platform.

The remainder of this paper is organized as follows. Section II introduces the existing CSI feedback framework in 5G systems, and associated AI-enabled studies. The framework of our developed platform is elaborated in Section III. Experimental results are presented in Section IV and conclusions are drawn in Section V.

II. 5G CSI Feedback Frameworks

We introduce the codebook-based CSI feedback and AI-enabled CSI feedback in this section. Codebook-based CSI feedback framework has been adopted in existing 4G and 5G cellular systems [11]. Codebook in the context of CSI feedback is a set of precoding matrices shared by the UE and the Base Station (BS). Under this framework, UE searches the best matching codeword for the estimated channel and feedbacks the index of the selected codeword to the BS. The BS acquires the corresponding precoding matrix by looking up the shared codebook with the feedback codeword index. This feedback is constrained by two factors: codebook resolution

This article is based upon work partially supported by InterDigital Inc. and the U.S. National Science Foundation under grants CNS-1925601.

ISBN: 978-3-903176-61-4

© 2024 International Federation for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

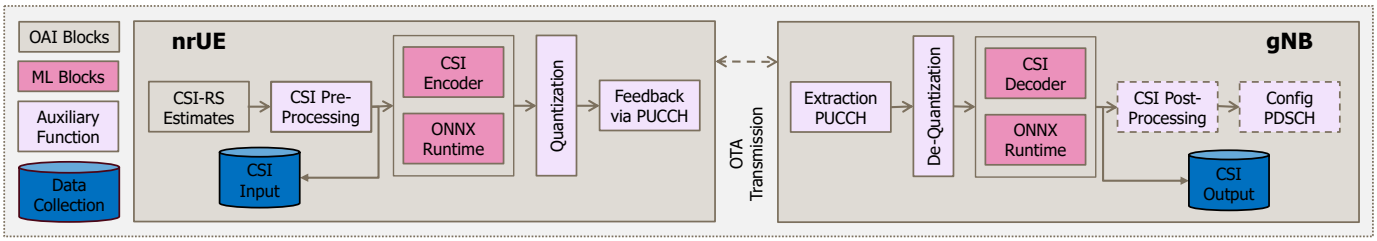

Fig. 1. Framework of End-to-end 5G Open RAN experiments. The light gray boxes are blocks/modules from OAI. The pink boxes are AI/ML modules (models or runtime). The light lavender boxes are auxiliary functions developed to support the running of AI/ML modules. The dotted lavender boxes are the modules beyond the scope of this paper. The blue boxes represent the collected data.

and codeword lookup algorithm complexity. In massive Multiple Input Multiple Output (MIMO) systems, the codebook size increases drastically and thus the feedback overhead, as well as the memory and computational complexity increase to maintain the feedback accuracy.

AI-enabled CSI feedback is one possible option to address these aspects. An autoencoder-based CSI compression model can accommodate this use case. In an autoencoder-based CSI compression framework, the NN encoder compressing the CSI into codewords is placed at the UE side and the NN decoder reconstructing the CSI is placed at the gNB side. Following this compression framework, we developed our real-time AI-enabled CSI feedback platform. To be specific, the raw channel estimates computed from the 5G protocol stack are pre-processed and then fed to a CSI encoder at the UE side. The gNB reconstructs the pre-processed channel by decoding the CSI feedback from the UE. The details of our platform are given in the next section.

III. AI-enabled CSI Feedback in Open RAN

In this section, we explain the framework for our real-time AI-enabled CSI feedback platform, which is depicted in Fig. 1. First, we introduce OAI, which is an Open RAN compliant 5G protocol stack implementation, and how to integrate CSI autoencoder into OAI codebase with the help of ONNX runtime. Then, we introduce how our CSI autoencoder is trained, including CSI dataset generation and the NN architecture. The CSI feedback performance is discussed at the end.

A. Integration of CSI Autoencoder into 5G Networks

1) OpenAirInterface 5G Networks: OAI [12] is an open-source Open RAN compliant reference implementation of 4G/5G Radio Access Network (RAN) and core network. OAI is designed to run on general-purpose x86 CPU together with Software-defined Radio (SDR). In our platform, we adopt OAI gNB as a 5G base station and OAI nrUE as a 5G UE to form an end-to-end 5G network. Both gNB and nrUE are running on x86 Ubuntu equipped with Universal Software Radio Peripheral (USRP) Radios. Note that we also use OAI 5G core network in the experiment. However, we omit it from the block diagram since it is unrelated to CSI feedback.

2) ONNX Runtime: ONNX [13] is an open format designed to represent DL and traditional ML models. Three benefits make ONNX runtime a good choice for real-time inference. *First*, ONNX defines a common set of DL/ML building blocks and a common file format, which enables its interoperability between different deep learning frameworks, such as TensorFlow, PyTorch, MATLAB, and so on. This feature allows AI developers to train models in one framework and deploy the models in another framework by ONNX model converting. That also makes it convenient to use and compare various pre-trained models from different frameworks. *Second*, ONNX runtime enables efficient and fast inference on various platforms, which is crucial for latency-aware 5G RAN applications. *Third*, x86 CPU-based ONNX runtime is lightweight (18MB in Ubuntu), which makes it convenient to be integrated with the OAI codebase.

3) Auxiliary Functions and CSI Autoencoder: To enable the inference of the CSI autoencoder within OAI 5G protocol stack, we first split the autoencoder into CSI encoder and CSI decoder. The encoder is at the UE side and the decoder is at the gNB side, as shown in the pink box of Fig. 1. Some auxiliary functions are developed to support the inference of encoder and decoder, as shown in the lavender box of Fig. 1.

At *UE side*, the raw CSI estimates are first pre-processed and then fed into the CSI encoder. Meanwhile, the processed CSI is saved as *CSI Input* data, as shown in the blue box at nrUE part of Fig. 1. The saved data would be used to calculate the reconstruction accuracy in Section. IV. The output of the CSI encoder is full-precision floating point numbers. They are quantized into N-bit integer codewords.[1] Those CSI codewords are finally feedback to gNB via Physical Uplink Control Channel (PUCCH).

At *gNB side*, the CSI codewords are first extracted from the PUCCH payload and then de-quantized into floating point numbers. With the fed floating-point codewords, the CSI decoder generates reconstructed CSI, which is saved as *CSI Output* data. The reconstructed CSI needs to be post-processed and finally used to configure Physical Downlink Shared Channel (PDSCH), like Modulation and Coding Scheme (MCS) and MIMO precoding matrix.

B. Dataset Generation and Autoencoder Architecture

1) Dataset: We use MATLAB 5G toolbox [14], [15] to generate a dataset for CSI autoencoder training. A 2×2 MIMO clustered delay line (CDL) D channel is simulated, where

[1]We will explore the influence of N in Sec. IV.

ISBN: 978-3-903176-61-4

the carrier frequency is 3.6GHz, delay spread is 300ns, and UE moving speed is 0.5 m/s. The 5G carrier is with 30KHz subcarrier Spacing and 106 Physical Resource Block (PRB), which corresponds to 40MHz bandwidth.

Given the above setting, the raw channel estimate from *one RX port within one slot* is a complex array with the size of $[1272, 14, 2]$, where $1272 = 12 * 106$ is the number of subcarriers for 106 PRB, 14 is the number of OFDM symbols within one slot, and 2 is the number of TX antennas. Since the channel can be considered to be constant within one slot, we average the estimates over 14 OFDM symbols and thus the estimate is with size $[1272, 2]$. To further reduce the size of channel estimates, channel estimates are sparsified by applying 2D discrete Fourier transform (DFT), which transforms the raw channel estimates in the spatial frequency domain into the angular-delay domain. In the delay domain, we extract the first 24 row where it contains the most non-zero value. We get channel estimates with size $[24, 2]$ so far.

Note that the channel estimates are complex and NN process real numbers in general. Thus, we convert channel estimates from complex to separate real and imaginary parts. Therefore, the pre-processed channel estimates are real matrices with the size of $[24, 2, 2]$. We generate 15K channel estimates in total, where 10K is used for training, 3K is used for validation, and 2K is used for testing. The above data generation and pre-processing procedures basically follow the steps in [2].

2) Autoencoder NN Architecture: We follow the autoencoder NN architecture in [2] and the example in [15]. In our work, the output of the encoder, named latent variables, is designed to be with size $[2, 1]$. The details of the NN architecture are depicted in Fig. 2. During inference, the output of the encoder is first quantized into codewords and then fed to the decoder via PUCCH, as shown in Fig. 1. For the training stage, the output of the encoder is directly fed into the decoder. In terms of complexity, the encoder has 240 parameters and 17.3K floating point operations (FLOPs). For the decoder, it has 3753 parameters and 24.3K FLOPs.

C. CSI Feedback Performance

The main challenges of CSI feedback are the requirements for high reconstruction accuracy and low feedback latency at gNB, given the limited uplink bandwidth of UE. Thus, the evaluation of a CSI feedback framework relies on three metrics: feedback overhead, reconstructed CSI accuracy, and inference latency. For *feedback overhead*, it is measured by the total number of feedback bits, which is a multiplication of the number of codewords and quantization bits. For *reconstruction accuracy*, it is measured by the normalized mean-square error (NMSE) between the input CSI and the reconstructed CSI. For *inference latency*, it is measured by the execution time of the encoder at the UE side and the decoder at the gNB side. The execution time is determined by the model complexity, AI/ML runtime, and hardware platform.

Note that the CSI feedback is finally used to configure PDSCH. In our above discussion, we only consider a reconstructed accuracy metric, which is the NMSE. Although the throughput of PDSCH is a key system metric worth investigating, it is however beyond the scope of this paper. Due to this, we do not discuss the details of CSI post-processing and PDSCH configuration, which are in the dotted box at gNB part of Fig. 1.

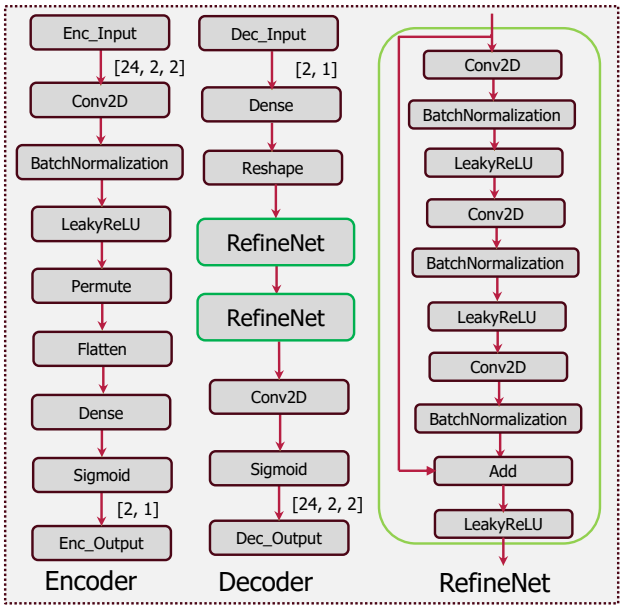

Fig. 2. CSI Encoder and Decoder NN Architecture.

IV. Experiments and Results

We conduct real-time experiments in Arena and Colosseum testbed. *Arena* is an indoor office testbed with radio transmitting over the office space via a non-mobility Line-of-Sight (LOS) channel. We use USRP X310 as Radio Unit (RU) of gNB and UE. Both gNB and UE are on the same type of host with 6-Core Intel E-2146G CPU, 32GB memory, and Ubuntu 22.04. The antenna's distance between gNB and UE is about 0.5 meter. We run 2×1 MIMO experiments on band n78 with 40MHz bandwidth.[2] In the experiments on *Colosseum*, we emulate a non-mobility single-tap channel model. The same type of host with 12-Core Intel E5-2650 CPU, 128GB memory, and Ubuntu 18.04 serves gNB and UE. The other experiment setting is the same as Arena. A portable demonstration setup (similar to Arena) and instantaneous monitored metrics are present in Fig. 3.

Accuray and Latency: We compare the CSI reconstruction accuracy under various codewords quantization bits, based on the experiment from the above two testbeds. The results are summarized in Table 1. The NSME increases significantly as the number of codeword bits is decreased. The feedback overhead is $16/14/12/10/8$ bits which corresponds to the $8/7/6/5/4$ bits codewords. The inference latency is summarized in Table 2. The pre-processing latency is mainly from the 2D DFT operation. The impact of the inference latency on

[2]We use 2×1 MIMO in our experiments since 2×2 MIMO is unstable with current OAI code.

ISBN: 978-3-903176-61-4

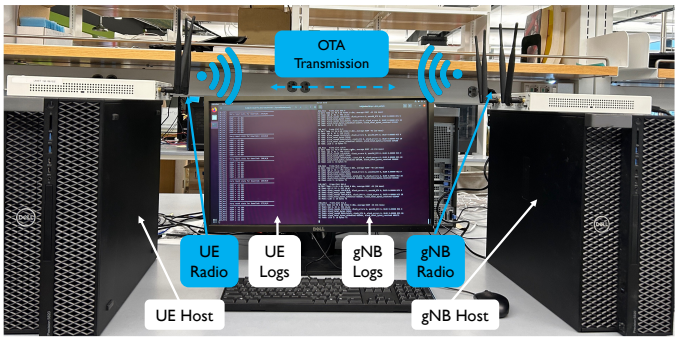

Fig. 3. A portable setup of our developed real-time OTA experiment platform.

other signal processing functions in OAI can be alleviated by using a separate thread for inference operations. Besides, the sub-ms latency is lower than the channel coherence time and thus it can be used to configure PDSCH.

Comparison to NR Standards: With the given experiments setting, the codebook-based CSI compression, implemented in OAI compliant to NR Release-15, achieve 7 bits CSI feedback overhead. However, NR codebook-based feedback overhead in massive MIMO systems increases to be even larger than AI-based feedback overhead. For instance, for a $32 * 2$ MIMO setting, and as shown in [16], the feedback overhead from the new codebooks proposed in NR Release-16 and Release-17 can be larger than 400 bits. Meanwhile, the AI-based feedback overhead can achieve the same or even lower overhead by tuning the compression ratio [2]. Thus, the benefits of AI-based CSI feedback are more significant in massive MIMO systems. A fair PDSCH throughput comparison between AI-based feedback and NR codebook-based feedback is meaningful but is out of the scope of this paper.

Table 1. NMSE (dB) of various CSI feedback codeword bits.

# Codeword bits	8	7	6	5	4
Arena	-17.60	-13.46	-7.35	-4.04	-2.20
Colosseum	-18.95	-13.54	-8.86	-7.64	-6.52

Table 2. Latency (ms) of pre-processing, encoder, and decoder.

	Pre-processing	Encoder	Decoder
Arena	0.228	0.059	0.105
Colosseum	0.247	0.104	0.283

V. Conclusion and Discussion

In this paper, we developed a real-time AI-enabled CSI feedback platform based on 5G Open RAN. In the developed platform, the 5G protocol stack is based on open-source OAI code. The AI/ML model inference is based on ONNX runtime which is a good intermediate representation to various AI/ML frameworks. An autoencoder-based CSI compression model is trained from MATLAB simulation data. The CSI autoencoder is deployed at Arena and Colosseum testbed. The experiment results, including feedback overhead, CSI reconstruction NMSE, and inference latency are present to demonstrate the functionality. Our current work shows the capability of adopting AI-enabled CSI feedback in Open RAN.

Note that our model is trained in full precision floating point numbers and is based on MATLAB CDL channel simulation data. The performance of our platform can be improved by the following two directions. First, a mixed-precision model can reduce the model complexity and inference time with little degradation in accuracy. This can be achieved by a mixed-precision training or post-training model quantization. Second, retraining or online learning according to the deployment channel scenarios could also increase the model's accuracy.

Many challenges remain to be investigated to demonstrate the feasibility and the deployment of two-sided AI/ML models in the Air Interface, which is the case of auto-encoders for CSI feedback studied in this paper. Some of these challenges include, for instance, scalability and generalization, practical multi-vendor deployments, and overall system performance measurement and monitoring.

References

[1] X. Lin, "Artificial Intelligence in 3GPP 5G-Advanced: A Survey," May 2023. [Online]. Available: https://arxiv.org/abs/2305.05092

[2] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive mimo csi feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.

[3] M. Chen, J. Guo, C.-K. Wen, S. Jin, G. Y. Li, and A. Yang, "Deep learning-based implicit csi feedback in massive mimo," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 935–950, 2022.

[4] Y. Cui, A. Guo, and C. Song, "Transnet: Full attention network for csi feedback in fdd massive mimo system," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 903–907, 2022.

[5] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[6] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[7] B. Azari, H. Cheng, N. Soltani, H. Li, Y. Li, M. Belgiovine, T. Imbiriba, S. D'Oro, T. Melodia, Y. Wang *et al.*, "Automated deep learning-based wide-band receiver," *Computer Networks*, vol. 218, p. 109367, 2022.

[8] J. Guo, C.-K. Wen, S. Jin, and X. Li, "Ai for csi feedback enhancement in 5g-advanced," *IEEE Wireless Communications*, pp. 1–8, 2022.

[9] L. Bertizzolo, L. Bonati, E. Demirors, A. Al-Shawabka, S. D'Oro, F. Restuccia, and T. Melodia, "Arena: A 64-antenna sdr-based ceiling grid testing platform for sub-6 ghz 5g-and-beyond radio spectrum research," *Computer Networks*, vol. 181, p. 107436, 2020.

[10] "Colosseum," Nov 2023. [Online]. Available: https://www.colosseum.net

[11] 3GPP TS 38.214, "Physical layer procedures for data," Tech. Rep., 5 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/17.01.00_60/ts_138214v170100p.pdf

[12] "Openairinterface," Nov 2023. [Online]. Available: https://openairinterface.org/

[13] ONNX Runtime developers, "Onnx runtime," 2023, version: 1.16.0. [Online]. Available: https://onnxruntime.ai/

[14] The MathWorks Inc., "5g toolbox," 2023, version: 2.6, R2023a. [Online]. Available: https://www.mathworks.com/products/5g.html

[15] ——, "Csi feedback with autoencoders," 2023. [Online]. Available: https://www.mathworks.com/help/5g/ug/csi-feedback-with-autoencoders.html

[16] Z. Qin and H. Yin, "A review of codebooks for csi feedback in 5g new radio and beyond," *arXiv preprint arXiv:2302.09222*, 2023.

ISBN: 978-3-903176-61-4