Next-Gen AI-on-RAN: AI-native, Interoperable, and GPU-Accelerated Testbed Towards 6G Open-RAN

Osman Tugay Basaran*†, Hammad Zafar†, Martin Kasparick†, Falko Dressler*, and Slawomir Stańczak*†
*School of Electrical Engineering and Computer Science, TU Berlin, Germany
†Fraunhofer Heinrich-Hertz-Institut, Berlin, Germany
{basaran, dressler}@ccs-labs.org, {hammad.zafar, martin.kasparick, slawomir.stanczak}@hhi.fraunhofer.de

***Abstract*—The evolution of 5G and the emergence of 6G networks demand advanced, artificial intelligence-enabled (AI-enabled), interoperable, and scalable solutions to overcome the ever-increasing complexity and performance requirements of next-generation wireless communication systems. To address these design challenges, we propose a Next Generation AI on Radio Access Network (Next-Gen AI-on-RAN) testbed an open-source platform that leverages NVIDIA Aerial RAN CoLab (ARC) and Aerial Software Development Kit (SDK) to enable virtualized Graphics Processing Unit (GPU) accelerated Layer 1 (L1) and Layer 2 (L2) processing, significantly enhancing computational efficiency and reducing latency. A novel aspect of this work is the co-location of training and inference workloads on the same NVIDIA GPU-centric RAN server, wherein the GPU is simultaneously used for real-time L1/L2 processing and AI model training for traffic prediction. Experimental analysis demonstrates that our design 32% reduced training time without sacrificing model accuracy which underscores the potential of GPU-accelerated AI-on-RAN for robust next-generation wireless networks.**

***Index Terms*—Virtualization, AI-on-RAN, 6G, xApps, GPU-Centric, O-RAN**

I. Introduction

As the telecommunications industry transitions from 4G to 5G [1] and begins to lay the groundwork for 6G, the complexity of wireless networks is rapidly increasing [2]. 5G networks have introduced significant advancements, such as enhanced mobile broadband, ultra-reliable low-latency communications, and massive machine-type communications, which together form the backbone of modern smart cities, autonomous vehicles, and industrial automation. However, these advancements also introduce new challenges in network design, management, and optimization, particularly when it comes to ensuring seamless interoperability between diverse network components and vendors.

The advent of Open Radio Access Network (O-RAN) [3] has introduced a new paradigm in network architecture, emphasizing interoperability, openness, and disaggregation of network functions [4]. In traditional RAN architectures, network components were often proprietary and tightly integrated, limiting flexibility but ensuring a level of performance consistency through highly optimized hardware. However, the transition to O-RAN [5], with its focus on vendor-neutral interfaces and open-source software, necessitates a more sophisticated approach to performance monitoring and troubleshooting. As a result, open-source and interoperable testbeds [6] have become essential tools in the development and deployment of next-generation wireless communication systems. Moreover, the integration of artificial intelligence (AI) and machine learning (ML) into network management and optimization is a critical aspect of these future wireless networks [7].

Despite these advancements, there is an increasing need to further optimize O-RAN networks to meet the rigorous performance and efficiency demands of modern telecommunications. A significant challenge for such virtualized open-source platforms is addressing the high computational complexity of Digital Signal Processing (DSP) for the Physical (PHY) layer, which consumes substantial computing resources when executed on general-purpose CPUs. Additionally, maintaining high-performance networks requires supporting intensive optimization algorithms capable of swiftly and accurately diagnosing issues related to sudden network anomalies in both the RAN and user equipment (UE), as well as providing solutions to mitigate and enhance network performance.

A promising approach to achieving this optimization is the integration of accelerators and specialized hardware designed to expedite these intensive operations. By utilizing such hardware, processing speeds can be significantly improved not only for RAN operations but also for executing AI-based optimization algorithms. This approach could enable an AI-powered O-RAN deployment that not only meets the demands for higher performance but also supports the scalability and sustainability of next-generation network infrastructures. To address these challenges, this work presents an innovative AI-on-RAN testbed as seen in Figure 1.

The core outcomes of our study are summarized as new contributions ("C") and new findings ("F") as follows:

- **C1.** We developed a customized eXtended application (xApp) catalogue, integrating the O-RAN Near Real-Time (Near-RT) RAN Intelligent Controller (RIC) with NVIDIA ARC-based RAN to enhance real-time network management.
- **C2.** This work presents the first instance in the literature where GPU infrastructure is used concurrently for both real-time L1/L2 layer network processing and the execution of an AI-driven traffic prediction model.
- **C3.** Our solution, which includes custom Key Performance Metric (KPM) and traffic prediction (TP) xApps, along with a comprehensive monitoring interface, enables efficient real-time diagnostics and AI-powered network optimization.
- **F1.** GPU-centric RAN reduced the AI model training time by approximately 32%, enabling faster model updates

Table I
COMPARISON WITH EXISTING GPU-CENTRIC RADIO ACCESS NETWORK TESTBEDS

Testbeds	O-RAN Split 7.2	NVIDIA Aerial SDK	OAI Support	O-RAN RIC and xApps Catalogue	Real-Time Monitoring
Aerial [8]	✓	✓	χ	χ	χ
Aerial AI-on-5G [9]	✓	✓	χ	χ	χ
X5G Testbed [10]	✓	✓	✓	χ	χ
Next-Gen AI-on-RAN	✓	✓	✓	✓	✓

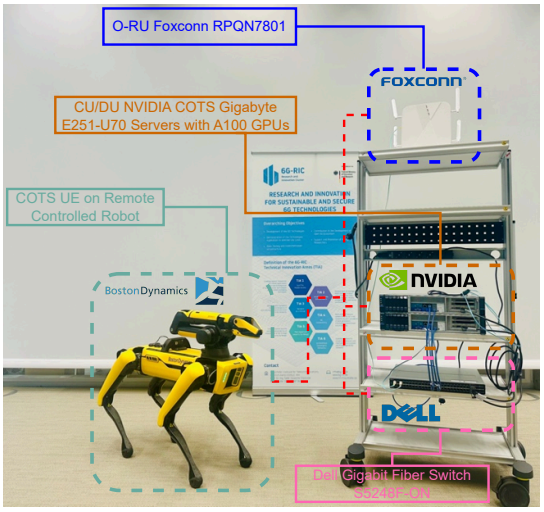

Fig. 1. Remote-Controlled User Equipment Connected Next-Gen AI-on-RAN testbed at HHI

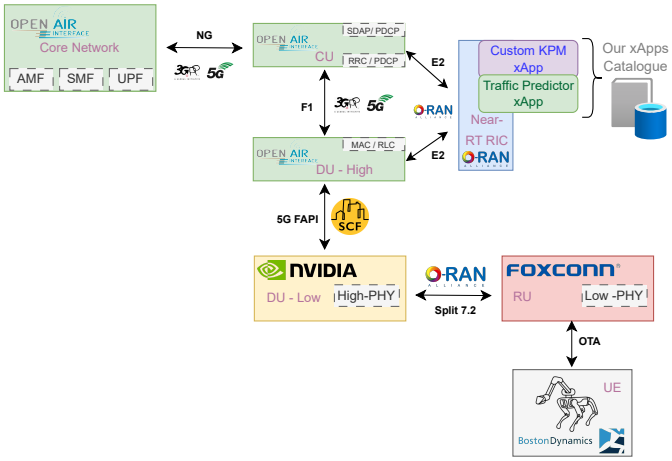

Fig. 2. End-to-End Architecture of our Multi-Vendor Next-Gen AI-on-RAN Testbed

and retraining cycles in AI-native networks.

F2. Our AI-based TP xApp model presents a 48.9% and 66.8% reduction in mean squared error (MSE) compared to the baseline models which indicates that our model's predictions are statistically closer to the actual downlink (DL) data rates, reflecting higher accuracy and reliability.

II. FUNDAMENTALS OF NEXT-GEN AI-ON-RAN TESTBED

NVIDIA, a leader in GPU technology, has introduced innovative solutions [8], [9] to accelerate the design and optimization of interoperable, virtualized Next-Gen RAN. However, since these studies by NVIDIA are reference works that reveal the tool, GPU-based L1/L2 processing research is still very new in 6G research. In [10], researchers focused on RF planning over the digital twin framework using NVIDIA ARC and OpenAirInterface network [11]. The X5G testbed provides concrete performance metrics such as DL and uplink (UL) rates in a multi-node deployment, offering insights into the practical capabilities of the testbed in a real-world setting. However, our proposed O-RAN RIC integration and xApp experiments are not included in this work, they defined it as future work. One-to-one feature comparison of our testbed with the aforementioned GPU-based testbeds can be seen in Table I.

Centralized and Distributed Units: Centralized Units (CUs) use OAI software which is ported onto the CPU in the testbed. NVIDIA ARC and Aerial SDK are designed to handle the computational demands of modern 5G networks and beyond, particularly in implementing Virtual Distributed Unit (vDU) layers in the RAN as shown in Figure 2. To process PHY layer data, the overall system consists of 8 ARC nodes working on Gigabyte E251-U70 servers. Broadcom PEX 8747 PCI switch was added to the system to manage interconnections. Mellanox ConnectX-6 Dx Network Interface Card (NIC) with 2 QFTP Ports is used for O-RAN frounthaul split 7.2 interface functionality [12]. To support NVIDIA Aerial SDK on PHY layer operations, the testbed is equipped with NVIDIA A100 GPU. Besides GPU computation power, NVIDIA ARC has a 24-core Intel Xeon Gold 6240R CPU and 96 GB of RAM. Compute Unified Device Architecture (CUDA) [13] is the parallel computing platform and programming model that underpins the Aerial SDK. Our system uses a 5G Functional API (FAPI) interface between DU-High and DU-Low which is specified by the Small Cell Forum (SCF) [14].

Foxconn RPQN 4T4R Radio Unit [15] and UEs: It is a 4T4R (4 Transmit, 4 Receive) radio unit, meaning it has four channels for both transmitting and receiving signals, which enables higher data throughput and better coverage compared to simpler configurations like 2T2R. It offers up to 40W of transmit power per channel, totaling 160W across its four channels, enabling extensive coverage and robust signal quality. The unit supports a wide range of frequency bands, including both sub-6 GHz and higher mmWave frequencies, making it versatile for various global network deployments. As UE, we used different 5G Commercial-off-the-shelf (COTS) UEs placed on the remote-controlled mobile robot.

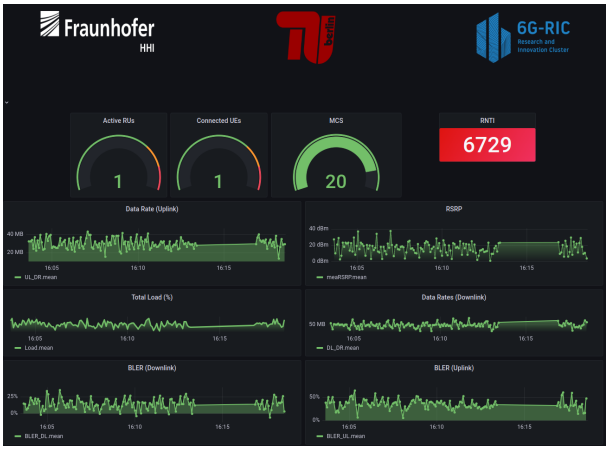

Fig. 3. Real-time KPI Monitoring User Interface via our Custom KPM xApp

O-RAN Near-RT RIC: O-RAN SC RIC E-release [16] is integrated in the testbed. We used a custom E2 Agent [17] to manage E2SM functionalities and enable interaction between Near-RT RIC and E2 nodes. E2SM-KPM is used for collecting and reporting key performance indicators (KPIs) and other measurements from the RAN to the Near-RT RIC. These real-time metrics allow the RIC to continuously monitor network performance. In turn, decisions on resource allocation and optimization are executed through the E2SM-RC (RAN Control) service model, which sends control policies back to the RAN. This integration of E2SM-KPM for monitoring and E2SM-RC for policy enforcement enables the Near-RT RIC to make informed, data-driven decisions for real-time network optimization.

Our xApps Catalogue: Our custom KPM monitoring xApp serves as a key component for enabling successful closed-loop control, leveraging E2SM-KPM for the real-time reporting of KPIs over the E2 interface. These KPIs include, but are not limited to, UL/ DL throughput (Th), block error rates (BLER), modulation and coding schemes (MCS), received power (RP), signal-to-noise-plus-interference-ratio (SINR), and E2 node load-related metrics. These data streams are stored within the Near-RT RIC and used by our TP xApp to forecast the future DL data rates of mobile users. TP xApp utilizes our custom KPM xApp to gather real-time traffic data, which it analyzes to learn traffic patterns. By leveraging these observations, TP xApp employs a Long Short-Term Memory (LSTM) network [18] to predict future traffic patterns. Both xApps utilize O-RAN-defined service models, the E2SM-KPM and E2SM-RC service model which enables end-to-end system optimization within the O-RAN framework.

Our Performance Monitoring User Interface: We have developed a performance monitoring user interface that integrates InfluxDB for data storage and Grafana [19] for data visualization as shown in Figure 3. This combination provides a powerful, flexible, and user-friendly solution for tracking and analyzing KPIs in real-time. InfluxDB serves as the backend time-series database, efficiently storing vast amounts of KPI data collected from network elements via the O-RAN E2 interface. This data is then visualized in Grafana, where customizable dashboards offer dynamic and interactive views of network performance metrics such as latency, throughput, and packet loss. Grafana's real-time monitoring, coupled with its alerting system, enables testbed users to quickly identify and respond to performance anomalies.

In our testbed, COTS UEs communicate with ARC gNBs, adhering to the fundamental O-RAN architecture divided into CU, DU, and RU. The Foxconn RUs are connected to the DU-low implemented using the NVIDIA Aerial SDK. Meanwhile, DU-high and CU, developed by the OAI are both containerized within a single container and communicate with DU-low via the 5G-FAPI interface. Data from the CU is then transmitted to the OAI core, which processes it while considering factors such as channel conditions, modulation schemes, and network congestion. Additionally, DU-high and CU are configured to interact with the Near-RT RIC through the E2 interface, enabling xApps to monitor network KPIs and deploy intelligent xApps to enhance network functionality and performance.

III. GPU-Accelerated AI-on-RAN

GPU acceleration is transformative for AI-based algorithms, especially in scenarios where real-time processing and high computational demands are critical. Therefore, integrating GPU acceleration into O-RAN architecture for AI-based xApp optimization marks a significant leap in network performance and efficiency. By leveraging GPU-based acceleration, we show that the same hardware traditionally used for baseband processing in the protocol stack can efficiently handle resource-intensive AI algorithms.

A. Traffic Prediction xApp Design

To properly define the TP xApp, we consider it as a functional module that processes real-time KPIs from the KPM xApp and predicts future DL data rates using an LSTM network. At each discrete time interval $t \in \mathbb{N}$, the KPM xApp collects a set of KPIs, which are then transmitted to the TP xApp. The feature vector constructed by the TP xApp is represented as:

$$\mathbf{x}_t = \{\text{Th}_t, \text{BLER}_t, \text{MCS}_t, \text{RP}_t, \text{SINR}_t\}$$

where n is the number of KPIs used. TP xApp maintains a sequence of the past L feature vectors to capture temporal dependencies:

$$\mathcal{S}_t = [\mathbf{x}_{t-L+1}, \mathbf{x}_{t-L+2}, \ldots, \mathbf{x}_t] \tag{1}$$

The LSTM network processes the sequence $\mathcal{S}_t$ to predict the future DL data rate. The LSTM model is defined as a function f_θ parameterized by weights θ:

$$\hat{y}_{t+1} = f_\theta(\mathcal{S}_t) \tag{2}$$

where $\hat{y}_{t+1}$ is the predicted DL data rate at time $t+1$. This predictive capability is crucial for proactive network management, allowing operators to allocate resources efficiently and avoid congestion.

B. Model Training and Inference on GPU-Centric RAN

GPUs, with their massively parallel architecture, are designed to handle thousands of simultaneous threads, making them exceptionally well-suited for the intensive computations required by AI algorithms, such as deep learning and neural networks. To clarify our GPU-Centric RAN operations, let us denote:

① T_p: Processing time required by the LSTM model to generate a prediction for a one-time step.
② T_{CPU}: Processing time when using a CPU.
③ T_{GPU}: Processing time when using a GPU.

Due to the parallel processing capabilities of GPUs, we have:

$$T_{\text{GPU}} = \frac{T_{\text{CPU}}}{k} \tag{3}$$

where $k > 1$ represents the acceleration factor provided by the GPU over the CPU. Assuming the xApp processes a batch of B sequences simultaneously to maximize GPU utilization, the total processing time for the batch T_{batch} is:

$$T_{\text{batch}} = T_{\text{GPU}} + T_{\text{overhead}} \tag{4}$$

where T_{overhead} includes data transfer times between the CPU and GPU memory. Then latency (L), the time elapsed from receiving the input sequence to producing the output prediction can be given as:

$$L = T_{\text{data\_prep}} + T_{\text{batch}} \tag{5}$$

where $T_{\text{data\_prep}}$ is the time taken to pre-process data. Similarly, the number of predictions made per unit time (Φ) can be calculated as:

$$\Phi = \frac{B}{T_{\text{batch}}} \tag{6}$$

To assess resource utilization on the GPU, we need the following:

① C_{GPU}: The computational capacity of the GPU in Floating Point Operations Per Second (FLOPS).
② D: The total computational demand of the LSTM model per prediction in FLOPs.

The utilization ratio (U) of the GPU is given by:

$$U = \frac{B \times D}{C_{\text{GPU}} \times T_{\text{batch}}} \tag{7}$$

To maximize throughput while minimizing latency, we can formulate an optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \Phi = \frac{B}{T_{\text{batch}}} \\ \text{Subject to} \quad & L \le L_{\max} \\ & U \le U_{\max} \\ & B \in \mathbb{N}^+ \end{aligned} \tag{8}$$

where maximum allowable latency $L_{\max}$ and maximum allowable GPU utilization (typically less than 1 to prevent overload) $U_{\max}$.

LSTM computations are parallelized on the GPU to accelerate matrix operations involved in the network's forward pass. The implementation of LSTM equations utilizes parallel matrix operations, as detailed below:

i) All weight matrices $\mathbf{W}_*$ and $\mathbf{U}_*$ are large matrices suitable for parallel processing.

1) Input Gate:
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$
2) Forget Gate:
$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$
3) Cell State Update:
$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$$
4) Cell State Calculation:
$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t$$
5) Output Gate:
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$
6) Hidden State Calculation:
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

ii) The element-wise operations, such as the activation functions σ and $\tanh$, as well as the Hadamard product $\odot$, are applied in parallel across vectors, ensuring efficient processing on the GPU.

To consider the data transfer overhead:

① T_{transfer}: The time required to transfer data between CPU and GPU memory.
② S: The size of the data being transferred (in bytes).
③ B_w: The bandwidth of the PCIe bus or interconnect (in bytes per second).

The transfer time can be calculated as:

$$T_{\text{transfer}} = \frac{S}{B_w} \tag{9}$$

Minimizing this overhead is crucial for enhancing overall performance. Furthermore, to deploy additional xApps effectively, memory and real-time processing constraints must be carefully managed; GPUs have a limited memory M_{GPU} and the batch size B should be selected such that:

$$B \times \text{Memory Footprint per Sequence} \le M_{\text{GPU}} \tag{10}$$

Deployed xApps must operate under real-time constraints to be effective in network management. The end-to-end delay D_{e2e} from data acquisition to prediction delivery should satisfy:

$$D_{\text{e2e}} = T_{\text{data\_acq}} + L + T_{\text{comm}} \le D_{\max} \tag{11}$$

where time to acquire data from the KPM xApp $T_{\text{data\_acq}}$, communication delay in delivering the prediction T_{comm}, and maximum allowable end-to-end delay $D_{\max}$. So our final TP xApp deployed on a GPU server can be formally defined as:

$$\hat{y}_{t+1} = \mathcal{F}_{\text{GPU}}(\mathcal{S}_t; \theta_{\text{GPU}}) \tag{12}$$

where LSTM model optimized for GPU execution $\mathcal{F}_{\text{GPU}}$ and model parameters stored and processed on the GPU θ_{GPU}.

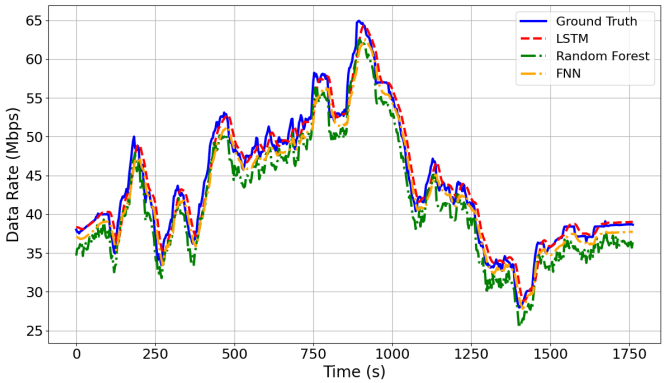

Fig. 4. Data Rate Prediction using LSTM-based TP xApp

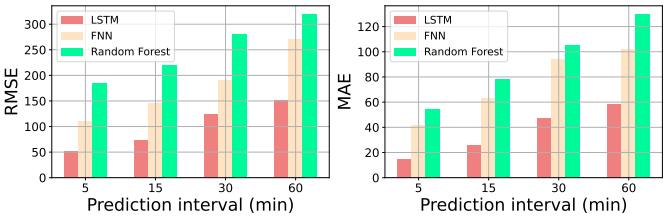

Fig. 5. Model Prediction Error within Different Time Intervals

IV. Experiments and Results

A. Benchmarking with Baseline Forecasting Models

To evaluate the performance of the LSTM-based model, we fine-tuned the hyperparameters for optimal results. The batch size was set to 64, and the model was trained over 500 epochs. The model is trained using the MSE loss function, which measures the average squared difference between predicted and actual traffic values. The model was trained on 24 sets of 30-minute intervals, which amounted to 12 hours of data. During training, the model's performance was continuously monitored using a validation dataset derived from the training data. The validation loss was used as a metric to adjust the model's hyperparameters and prevent overfitting. The Adam optimizer is employed with an initial learning rate of 0.01. Adam is chosen for its efficiency in handling sparse gradients and adaptive learning rates. To enhance training efficiency, an early stopping strategy with patience of 5 epochs is implemented. If the training loss does not decrease for 5 consecutive epochs, the learning rate is reduced by a factor of 0.1. This approach helps prevent overfitting and ensures convergence. After training, the model was tested on a separate dataset comprising 4 sets of 30-minute intervals, totaling 2 hours of data. The purpose of this testing phase was to evaluate the model's generalization capability on unseen data. The LSTM-based approach, while computationally intensive during the training phase, offers superior prediction accuracy compared to simpler methods like the minimum distance method. Once trained, the LSTM model can deliver precise traffic forecasts with only a marginal increase in computational complexity during inference.

In Fig. 4, we evaluate the prediction performance of the LSTM-based TP xApp and compare it with alternative

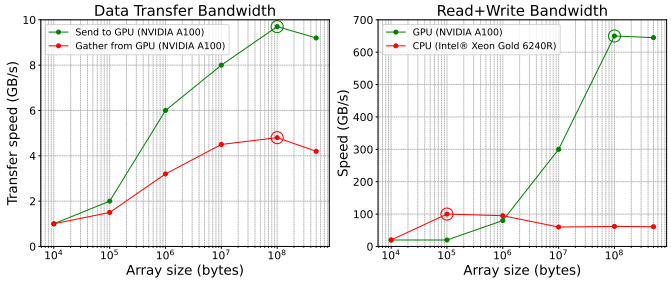

Fig. 6. Performance Comparison for Memory Intensive AI-on-RAN Operations

Table II
Performance comparison between CPU and GPU centric servers for training the LSTM model under different training configurations

Processor Type	Epoch = 300	Epoch = 400	Epoch = 500
CPU (Intel Xeon Gold)	4055s	5900s	6980s
GPU (NVIDIA A100)	3125s	4241s	5286s

methods: using Random Forest (RF) [20] and the Feed-foward Neural Network (FNN) [21]. For the RF model, we configured 200 trees, striking a balance between performance and training time, while limiting the maximum tree depth to 10 to mitigate overfitting and allow the model to effectively learn meaningful patterns in the data. The FNN was designed with three hidden layers containing 64, 32, and 16 neurons, respectively, utilizing a ReLU activation function for hidden layers and a linear activation for the output layer. In the figure, the solid line represents the actual DL data rates observed during the test period, while the dashed line illustrates the predicted values from the different models.

As depicted in Fig. 4, the predicted DL data rates from the LSTM-based TP xApps align closely with the ground truth, demonstrating the model's effectiveness in capturing temporal variations and accurately forecasting future rates. In contrast, the other methods demonstrate lower prediction accuracy in terms of error rate. The MSE for the LSTM model is 0.90, while the FNN has an MSE of 1.76. The RF model exhibits the highest MSE at 2.71, making it the least accurate among these models. This represents a 48.9% reduction in MSE compared to the FNN and a 66.8% reduction compared to the RF model. LSTM's ability to model temporal dependencies within the traffic data allows it to capture complex patterns and fluctuations more effectively than models lacking recurrent architectures. Statistical tests, such as paired t-tests on the prediction errors, confirmed that the improvements offered by the LSTM are statistically significant ($p-value<0.01$), validating its efficacy in enhancing traffic forecasting accuracy. Also when model prediction errors are evaluated in Fig. 5, the LSTM model demonstrates superior predictive performance across various time intervals, as evidenced by lower root mean square error (RMSE) and mean absolute error (MAE) values compared to FNN and RF models. At the 5-minute interval, relative improvement translates to a 33% reduction in RMSE, 40% in MAE compared to FNN, and a 50% reduction in both RMSE and MAE compared

to RF. Such reductions indicate that LSTM significantly mitigates prediction errors. Variance in RMSE and MAE is crucial for understanding consistency. Lower variance in these metrics for LSTM suggests that its predictions are more stable and less prone to extreme errors, especially over longer intervals.

B. GPU versus CPU for AI-on-RAN Operations

When analyzing Fig. 6, it becomes clear that utilizing GPUs for both data transfer and memory operations leads to significantly reduced latencies, facilitating real-time processing for RAN L1/L2. The GPU's high sending speed of 10 GB/s and gathering speed of 4 GB/s yield a data transfer latency of just 35 ms for a 10^8 byte dataset, which is crucial for real-time data offloading in AI-on-RAN. With $6.5\times$ faster read+write speed on GPUs compared to CPUs, the GPU can process large datasets in a fraction of the time, enabling quicker AI inference results. Reduced latency of processing on GPUs enables faster updates and responses to changes in the RAN environment. This makes the AI models more effective in adapting to real-time changes in the wireless channel or user behavior, directly contributing to better Quality of Service (QoS) and Quality of Experience (QoE) towards 6G. For example, if decisions need to be made within a 50 ms window, using the GPU (0.154 ms) allows for more computation and adaptation compared to the CPU (potentially taking 1 ms or more with slower read/write operations). Also, execution times of the proposed LSTM-based prediction model are compared across different platforms in Table II to demonstrate the benefits of incorporating GPU acceleration into the O-RAN architecture for AI-based xApp optimization. For this comparison, we used an Intel CPU with 10 cores running at 2.7 GHz and 64 GB of RAM, alongside an NVIDIA A100 GPU. The training times for machine learning models varied significantly between the two platforms. For instance, the neural network used for data rate prediction required only 5286 seconds to train on the GPU, compared to 6980 seconds on the CPU. Similarly, substantial reductions in training time were observed across various configurations. These results highlight the clear advantages of GPU-based systems, particularly when handling larger datasets and complex models, where the efficiency gains lead to significant time savings. Leveraging GPU capabilities in AI-on-RAN scenarios can ultimately support the vision of ultra-reliable and low-latency communication (URLLC) in 6G networks.

V. Conclusion

Our Next-Gen AI-on-RAN testbed delivers significant improvements in computational efficiency for 6G networks. It provides faster data transfer speeds and substantially quicker memory operations on GPUs compared to traditional CPU-based approaches, reducing latency and enabling real-time responsiveness crucial for Next-Gen RAN applications. By running a computationally efficient PHY layer alongside AI models directly on the GPU, the testbed minimizes overhead and streamlines processing. Additionally, the integration of a versatile xApp catalog; KPM, TP xApps, and a real-time monitoring user interface—introduces a robust solution for detailed network diagnostics and AI-powered, GPU-accelerated network performance optimization.

Acknowledgment

This work has been funded by the German Federal Ministry of Education and Research (BMBF, Germany) as part of the 6G Platform under Grant 16KISK050, as well as 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K.

References

[1] M. Agiwal, H. Kwon, S. Park, and H. Jin, "A Survey on 4G-5G Dual Connectivity: Road to 5G Implementation," *IEEE Access*, vol. 9, pp. 16 193–16 210, 2021.

[2] H. Viswanathan and P. E. Mogensen, "Communications in the 6G Era," *IEEE Access*, vol. 8, pp. 57 063–57 074, 2020.

[3] 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*, pp. 1376–1411, 2023.

[4] A. Bhattacharyya, S. Ramanathan, A. Fumagalli, and K. Kondepu, "Towards Disaggregated Resilient 5G Radio Access Network: A Proof of Concept," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, 2023, pp. 396–401.

[5] M. Polese, M. Dohler, F. Dressler, M. Erol-Kantarci, R. Jana, R. Knopp, and T. Melodia, "Empowering the 6G Cellular Architecture With Open RAN," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 245–262, 2024.

[6] A. Gabilondo, Z. Fernandez, Á. Martín, R. Viola, M. Zorrilla, P. Angueira, and J. Montalbán, "5G SA multi-vendor network interoperability assessment," in *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, IEEE, 2021, pp. 1–6.

[7] S. Zhang and D. Zhu, "Towards artificial intelligence enabled 6G: State of the art, challenges, and opportunities," *Computer Networks*, vol. 183, p. 107 556, 2020.

[8] A. Kelkar and C. Dick, "Aerial: a GPU hyper-converged platform for 5G," ser. SIGCOMM '21, Virtual Event: Association for Computing Machinery, 2021, pp. 79–81.

[9] A. Kelkar and C. Dick, "NVIDIA Aerial GPU Hosted AI-on-5G," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 64–69.

[10] 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 *2nd Workshop on Next-generation Open and Programmable Radio Access Networks (NG-OPERA)*, Vancouver, BC, Canada, May 2024.

[11] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G Era," *Computer Networks*, vol. 176, p. 107 284, 2020.

[12] O-RAN Alliance, "O-RAN working group 4 (open fronthaul interfaces wg) control, user and synchronization plane specification," Techreport O-RAN.WG4.CUS.0-R003-v12.00, Jun. 2022.

[13] NVIDIA. "CUDA Toolkit." (2024), [Online]. Available: https://developer.nvidia.com/cuda-toolkit. (accessed: 10.08.2024).

[14] Small Cell Forum, "5G FAPI: PHY API Specification," Techreport 222.10.04, Oct. 2021.

[15] Foxconn. "Foxconn RPQN." (2024), [Online]. Available: https://fcc.report/FCC-ID/2AQ68RPQN7801/5573870.pdf. (accessed: 10.08.2024).

[16] O-RAN SC, *OSC Near Realtime RIC*, https://wiki.o-ran-sc.org/display/RICP/2022-05-24+Release+E, Accessed: 04-Aug-2024, 2024.

[17] E. Moro, M. Polese, A. Capone, and T. Melodia, "An Open RAN Framework for the Dynamic Control of 5G Service Level Agreements," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN 2023)*, Dresden, Germany, Nov. 2023, p. 6.

[18] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM cells and Network Architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[19] Grafana. "Grafana Monitoring." (2024), [Online]. Available: https://grafana.com/docs/. (accessed: 10.08.2024).

[20] S. J. Rigatti, "Random Forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.

[21] G. Bebis and M. Georgiopoulos, "Feed-forward Neural Networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, 1994.