Ericsson Technology Review

#4, April 2026

When AI has no time to think: real-time inference in radio access networks

Charting the future of innovation

Ericsson Technology Review — Charting the future of innovation | #4, April 2026

When AI has no time to think: real-time inference in radio access networks

Authors:
Pablo Soldati, Cristian Tatino, Burak Demirel, Jacek Wszolek, Chakri Padala

Real-time radio access network functions operate on microsecond-level deadlines that leave artificial intelligence with no time to think. The feasibility of inference under these conditions hinges not on increased hardware capability alone, but on carefully aligning model complexity with worst-case end-to-end execution budgets and compute platform capacity.

ISSN 0014-0171 284 23-3438 | Uen

Ericsson, SE-164 83 Stockholm, Sweden
Phone: +46 10 719 0000

Artificial intelligence (AI) is quickly becoming a strategic catalyst for enhancing performance, operational efficiency and autonomy in next-generation radio access networks (RANs).

Across the RAN protocol stack, researchers are exploring AI-based methods for tasks ranging from signal detection, channel estimation and beamforming in the physical layer (L1) to user and resource scheduling in the data link layer (L2), as well as traffic steering, load balancing and self-optimization of functions in the network layer (L3). However, deploying AI to replace real-time RAN functionalities poses challenges that fundamentally distinguish AI-for-RAN from more conventional AI application domains.

Real-time radio access network operations: understanding the requirements

Unlike classical AI applications in cloud-based services or offline analytics, where inference latency can often be traded against model complexity or accuracy, real-time RAN operations are governed by hard timing constraints. This is particularly true for most L1–L2 functionalities, which operate at timescales of a few tens of microseconds to align with the physical-layer transmission timeline, whereas higher-layer RAN functions may tolerate longer decision times.

Under these regimes, AI models have no time to think during inference. Model execution must occur within a fixed and extremely short time budget, leaving no opportunity for iterative computation, adaptive complexity or latency-accuracy trade-offs. The worst-case end-to-end (E2E) inference path is therefore not merely a performance metric, but a system-level requirement that determines whether AI can be deployed at all.

At the same time, AI-native RAN functions must operate reliably across highly heterogeneous and rapidly varying conditions, including diverse deployments, configurations, mobility patterns, traffic characteristics and radio environments [1]. Achieving such robustness typically calls for expressive models — often realized through larger parameter counts and complex model architectures — which translates into higher computational demand at inference time.

Deploying these models in real-time RAN systems, however, is challenging: inference must meet microsecond-level deadlines while operating under tightly provisioned compute resources, which limits the complexity of deployable model architectures. This creates a fundamental tension between the need for expressive, generalizable AI models and the stringent real-time latency and resource constraints of RAN platforms. Integrating AI into operational RANs therefore requires more than improved learning performance — it also requires engineering intelligence that can operate reliably under extreme latency and compute constraints.

Terms and abbreviations

AI – Artificial Intelligence | **CPU** – Central Processing Unit | **E2E** – End-to-End | **FP** – Float Precision | **GPU** – Graphics Processing Unit | **L1** – Physical Layer (Layer 1) | **L2** – Data Link Layer (Layer 2) | **L3** – Network Layer (Layer 3) | **LA** – Link Adaptation | **MIMO** – Multiple-Input, Multiple-Output | **ML** – Machine Learning | **MLP** – Multi-Layer Perceptron | **ms** – Millisecond(s) | **NR** – New Radio | **RAN** – Radio Access Network | **RL** – Reinforcement Learning | **SL** – Supervised Learning | **TTI** – Transmission Time Interval | **UE** – User Equipment | **µs** – Microsecond(s)

Deploying AI in radio access networks: real-time inference constraints

Most contemporary AI applications such as conversational agents and large-scale reasoning systems tolerate inference latencies from hundreds of milliseconds (ms) to seconds and rely on abundant, elastic compute and memory resources. **Figure 1** illustrates that RAN systems operate under entirely different conditions.

Latency-critical artificial intelligence

Any application of AI in RANs inherits the execution-latency budget of the functionality it is intended to replace. While these requirements are relatively relaxed for high-level network orchestration and management tasks, they become exceptionally stringent for real-time L1–L2 functions that directly support over-the-air transmissions, as shown in Figure 1.

The combined operations of physical and data layers must execute within the boundary of a transmission time interval (TTI), which in 5G New Radio (NR) ranges from 1ms down to 62.5µs depending on the physical layer numerology [2], while leaving time for data preparation, signal-to-resource mapping and transmission.

For many L1 functionalities, the execution-latency budget is constrained to the duration of a single orthogonal frequency-division multiplexing (OFDM) symbol — approximately 30µs for mid-band deployments and as low as 4µs-8µs in high-band scenarios. Other L1 functionalities can operate on a slot-duration basis, which with the 5G NR numerology can amount to 50µs and 400µs for high- and mid-band, respectively. Similarly, L2 functionalities share an aggregate execution window of roughly 50µs for high-band and about 200µs–300µs for mid-band deployments, as summarized in the table shown in **Figure 2**.

Despite these extremely short windows, L1–L2 processing involves computationally intensive operations. For example, user and resource scheduling in L2 are complex combinatorial optimization problems.

To accommodate these compute-heavy tasks within the overall timeline, less demanding procedures such as link adaptation (LA) are deliberately allocated only a small fraction of the available execution window. As a result, LA must execute within a few tens of microseconds. This window must cover all scheduled user equipment (UE) across all radio cells served by a baseband unit, as shown in Figure 2.

Therefore, when AI replaces an L1–L2 functionality, the full E2E inference execution path — including fetching and transforming input data, memory movement, any model-loading overhead and model inference latency itself — must align with the execution-latency budget allocated to the native RAN function.

Compute-budget-constrained artificial intelligence

The execution-latency budget of real-time RAN functions imposes a tight compute budget, defined as the maximum number of computations that can be executed within the available time window. The effective budget per problem instance fundamentally depends on the efficiency, parallelism and overheads of the underlying compute platform.

RAN compute platforms are traditionally purpose-built to support L1, L2 and L3 real-time operations and are, therefore, highly optimized around tightly pipelined, deterministic workflows. They typically integrate a mix of general-purpose processing elements — such as a central processing unit (CPU) and digital signal processing cores — alongside specialized accelerators, such as an application-

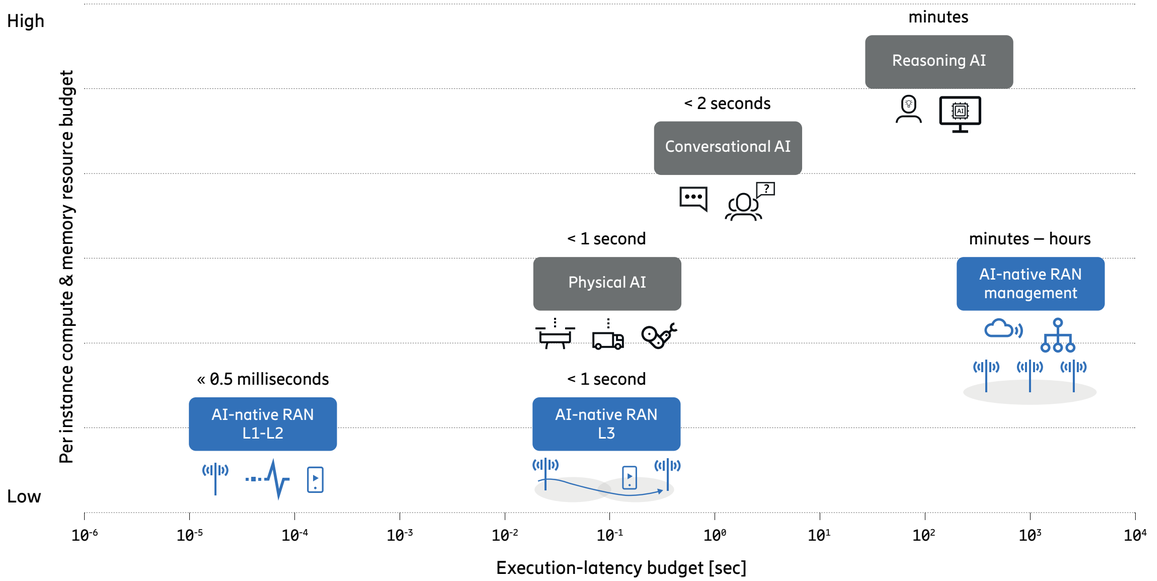

Figure 1: Comparison of execution-latency and compute budgets in AI-for-RAN versus traditional AI applications

Network functionality	Execution-latency budget	
	Mid-band	High-band
L1 - Symbol related	≈30µs	≈4µs-8µs
L1 - Slot related	≈400µs	≈50µs-100µs
L2 - Aggregate	≈200µs-300µs	≈50µs
L2 - LA	≈10µs-30µs	≈5µs

Figure 2: Execution-latency budget for L1–L2 operation under different 5G NR physical layer numerologies

specific integrated circuit and field-programmable gate array, designed to offload frequently executed, computationally intensive kernels, including fast Fourier transforms, detection, decoding and precoding. These accelerators achieve high compute throughput by exploiting highly specialized dataflows and memory hierarchies.

AI inference shares certain characteristics with these computationally intensive kernels and can, in principle, benefit from execution on traditional RAN accelerators. General-purpose AI accelerators such as graphics processing units (GPUs) may offer higher peak throughput and improved programmability for model inference. However, they can also introduce non-negligible overheads related to data movement, accelerator access and kernel invocation. Under tight real-time constraints, these overheads directly reduce the effective compute budget available for inference.

Figure 3 illustrates the trade-off between inference latency and platform overheads for different model sizes and compute platforms, relative to the typical execution-latency budget of LA in L2, assuming 32-bit float precision (FP32) and various batch sizes.

Real-time AI inference in the RAN is not primarily a hardware-scaling problem, but a model-design problem.

For small AI models the GPU overheads dominate inference latency, regardless of batching. In this regime, CPU-based inference often achieves lower latency, well within the L2 execution-latency bounds, despite lower peak throughput. However, CPU-based inference does not scale well with increasing batch size, even with small models.

GPU-based inference becomes advantageous with larger models or larger batches, when the computational demand is sufficient to amortize the platform overheads. Crucially, with even moderately larger AI models (0.5-1M params), inference latency alone exceeds the execution-latency budget of LA on both compute platforms.

Reconciling the gap between model expressiveness and real-time feasibility

Mitigating the tension between expressive AI models needed for robust generalization across heterogeneous and dynamic RAN environments and the strict execution-latency budgets of real-time RAN operations requires careful design choices.

A natural response is to integrate more powerful AI accelerators, such as GPUs, into RAN compute platforms. However, doing so can significantly increase both the capital and operational costs of RAN products, as well as overall energy consumption. More importantly, Figure 3 underscores that inference feasibility under real-time RAN constraints is governed not by peak compute throughput alone, but by microsecond-level execution budgets and platform-specific overheads. Crucially, these results show that even with state-of-the-art AI accelerators, inference for mildly expressive models (≈1M parameters) largely exceeds the execution-latency budget of most L1-L2 RAN functionalities. Therefore, executing inference within the TTI timeline remains challenging even with GPUs.

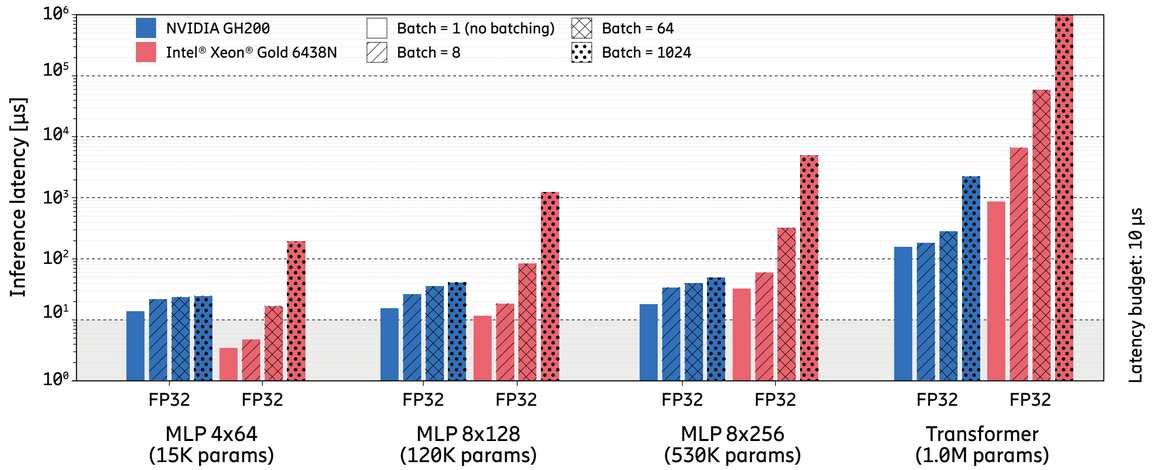

Figure 3: Worst-case inference latency versus model size and compute platform, compared with the microsecond-level execution-latency budget for L2 link adaptation. Note: All benchmarks performed on Intel Xeon Gold 6438N were executed on a single CPU core.

Another option is to relax the execution-latency budget by decoupling inference from the physical-layer transmission timeline, thereby extending the available execution window. While this can be viable for some time-critical L1–L2 functions, the additional execution headroom is typically limited to a few tens or hundreds of microseconds. For example, for LA in a downlink-heavy time-division duplex (TDD) configuration with a single uplink slot per radio frame, inference can be triggered upon receiving fresh UE feedback and reused across subsequent downlink slots. Nevertheless, LA inference values must still be available before the next downlink transmission, which ultimately limits the effective extension of the execution window.

Batch inference offers a different lever: by aggregating multiple inference requests, fixed overheads can be amortized and hardware utilization improved. However, batching is feasible only when inference outcomes are independent across intermediate decisions. This condition holds in certain scenarios, such as L1 signal detection and decoding, but not in tightly coupled or sequential tasks like multi-user MIMO scheduling and LA, where intermediate decisions shape the inputs needed for subsequent inference outcomes.

Taken together, accelerator integration, deferred inference and batching can mitigate specific deployment challenges, but they cannot provide a universal solution. When AI replaces real-time RAN functionalities, feasibility ultimately depends on whether the full worst-case end-to-end execution path fits within the application's time budget and the compute limits of the underlying platform.

These limitations point to a more fundamental conclusion: real-time AI inference feasibility in the RAN is not primarily a hardware-scaling problem, but a model-design problem. This motivates explicit model dimensioning for real-time execution as a foundational design principle, rather than relying solely on increased compute capability or system-level workarounds to accommodate larger models.

Importantly, model dimensioning does not imply training small models with limited expressiveness. Rather, it decouples learning capacity from deployment feasibility: large, expressive models can be trained to capture RAN complexity, while compact models are deployed for real-time execution. The central challenge is therefore to systematically transfer the robustness and generalization capabilities of high-capacity models into latency-feasible implementations without degrading performance — a problem that naturally leads to structured approaches such as model distillation.

Knowledge and policy distillation

Model distillation [3] is a general framework for knowledge transfer in which a student model learns to reproduce the behavior of one or more teacher models through imitation rather than direct supervision. The idea was introduced for compressing models [3] and subsequently formalized as knowledge distillation in supervised learning (SL) [4] and extended to reinforcement learning (RL) as policy distillation [5].

In knowledge distillation, a student model is trained to match the softened output distribution of a more expressive teacher by minimizing a divergence loss [4]. This makes it possible to compress large models into smaller ones while preserving accuracy. Policy distillation applies the same principle to RL: a student is trained to imitate a teacher's policy over environment states, with applications to model compression and multi-task knowledge transfer and unification allowing the consolidation of multiple specialized teacher policies into a single generalist policy.

A key design dimension of distillation distinguishes between single teacher and multi-teacher. Single-teacher distillation involves training a student to replicate a single expert and is commonly used for model compression. Multi-teacher distillation aggregates knowledge from multiple specialized teachers into a single student [6]. This setting has been extensively studied in multi-task learning, where multiple teachers specialized to different environments or tasks are distilled into a single multi-task policy capable of operating across diverse tasks or environments [7, 8, 9].

Various forms of model distillation can be applied to dimension AI models for RAN functions.

Dimensioning artificial intelligence for radio access networks

Various forms of model distillation can be applied to dimension AI models, replacing real-time L1—L2 RAN functions and meeting the associated execution-latency and compute budgets.

Supervised learning in RAN applications

SL applications span a wide range of L1-L3 functionalities, from channel estimation to predictions of traffic demand, resource utilization, cell load and mobility. Because SL model training does not require the model to interact with the live RAN environment, the constraints on compute, memory, and latency apply only at inference time, once the model is deployed. The separation between training and inference provides flexibility: high-capacity models can be trained offline, unconstrained by runtime limits.

Several distillation strategies are appropriate in this setting to transfer the generalization capability of a high-capacity teacher into a compact student model dimensioned to satisfy the execution-latency and compute budgets of the RAN application. Single-teacher distillation can compress a large model into a smaller student. Alternatively, multi-teacher distillation can be used to combine multiple specialized teachers trained on different data from distinct RAN conditions into a smaller unified student model. This enables the construction of compact models that remain robust across heterogeneous and dynamic RAN environments, while respecting the strict execution-latency and compute budgets imposed by real-time RAN operations.

Reinforcement learning in RAN applications

Reinforcement learning can be broadly categorized into offline and online settings. Offline RL learns solely from pre-collected datasets. In this context, policy distillation follows principles analogous to supervised learning: the offline RL training phase can leverage large-capacity models unconstrained by runtime limits, whereas the subsequent distillation phase dimensions the student to satisfy the compute and latency bounds of the target RAN function.

In contrast, policy evaluation during online RL training requires the model to continuously interact with the environment to generate new training data samples. Consequently, online RL in operational RANs requires models to remain latency-feasible throughout the training phase. This constraint limits the use of arbitrarily expressive models in an online RL setup. Instead, model dimensioning must be explicitly integrated as a core design aspect to ensure that every policy iteration satisfies real-time feasibility requirements.

Also in this case, model dimensioning can be achieved through policy distillation by decoupling policy learning from real-time policy execution. However, depending on how the RL process generates training data, policy distillation can operate in offline or online modes. In offline distillation, one or more teachers are first pre-trained through interactions with the environment and then used to generate distillation trajectories for training the student. In online distillation [5], the teacher continues to interact with and learn from the environment, while the student is continuously distilled from the teacher to track its evolving policy using newly generated trajectories [6, 10, 11]. Therefore, both approaches require the teacher to interact with the environment.

Crucially, in latency-critical RAN systems, neither offline nor online distillation can assume the presence of an unconstrained, high-capacity teacher operating in the live control loop and compress it after the fact. In this setting, any teacher used for data generation must satisfy the execution-latency and compute bounds of the target RAN function. Therefore, distillation must be integrated into the training architecture from the outset by enforcing latency-feasible model dimensioning throughout online RL learning.

Motivated by this observation, we propose two complementary paths for dimensioning models for online RL applications in real-time RAN operations. The first focuses on policy aggregation rather than compression [12]. It relies on multi-teacher offline distillation, where each teacher is pre-dimensioned to meet the available inference execution-latency budget and trained for a specific RAN condition (distinct mobility regimes or deployment topologies, for example). Although this path uses small teachers with limited expressiveness, offline multi-teacher policy distillation enables the fusion of these specialized behaviors into a single, more general student policy [12].

The second path is latency-aware online distillation, in which a large-capacity teacher is trained centrally using data generated by multiple actors across the network, learning a broadly generalized policy. Smaller students, dimensioned for the RAN platform, are continuously distilled to track the teacher's evolving policy during training following an online distillation framework, and are deployed (instead of the teacher) to distributed actors embedded in RAN nodes to interact with the environment. This strategy maintains latency-feasible inference through model dimensioning from the outset and produces a student model that inherits the generalization ability of the expressive teacher.

We propose two complementary paths for dimensioning models for online RL applications in real-time RAN operations.

A numerical example of a latency-aware online distillation process

To illustrate how latency-aware distillation can be applied in practice under realistic RAN constraints, we consider an RL-based LA [13] example. **Figure 4** demonstrates the feasibility of dimensioning AI models for online RL training under stringent RAN latency and compute constraints. It summarizes the performance achieved when RL models are tailored to the execution-latency and compute budgets of LA on a 5G RAN platform using the proposed latency-aware online distillation process.

The setup employs an eight-layer multi-layer perceptron (MLP) teacher model (≈115k parameters), whose inference latency exceeds the real-time execution budget of LA, alongside three progressively smaller student models, the smallest consisting of three layers with 32 neurons each (≈3.5k parameters). These student models are executable on a CPU platform and compliant with the LA execution-latency bounds identified in Figure 3. For comparison, we also trained a control model of identical size to the smallest student using a standard RL procedure, illustrating the limitations of directly training low-capacity policies.

The teacher is trained using distributed RL architecture [13], which combines distributed policy evaluation with domain randomization to enhance robustness to deployment variability, traffic fluctuations and user dynamics in RAN environments. During training, a student model dimensioned to RAN constraints is periodically distilled and deployed to distributed actors for data generation, ensuring that all online policy evaluation remains within RAN execution-latency limits.

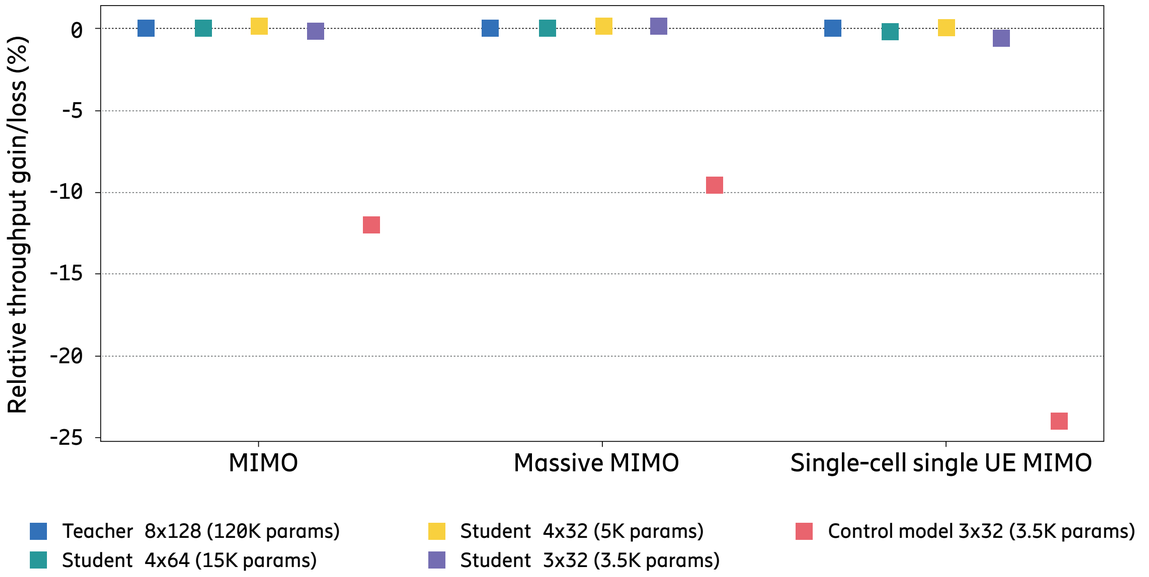

Figure 4: Model dimensioning using a latency-aware online distillation process – performance and generalization retention

The results shown in Figure 4 indicate that latency-aware online distillation produces compact student models that closely preserve the teacher's performance and generalization ability, while satisfying the stringent real-time LA latency and compute budgets of the target RAN platform. Generalization is evaluated on three unseen benchmark deployments, demonstrating that the distilled students consistently match the teacher across all scenarios. In contrast, directly training low-capacity models results in a 10 to 25 percent performance loss depending on the test environment, underscoring the necessity of distillation for producing RAN-compliant yet robust RL policies.

Two key conclusions can be drawn from the results presented in Figure 4:

1. Deployment constraints need not limit training model size and expressiveness, provided that a high-capacity teacher is distilled into real-time-capable students.

2. AI inference in latency-critical RAN operations cannot be achieved by hardware scaling alone; it requires principled, constraint-aware methods to align model complexity demands with worst-case E2E execution budgets and platform constraints.

Conclusion

Integrating artificial intelligence (AI) into real-time radio access networks (RANs) presents unique challenges because inference must operate within microsecond-level execution bounds and tightly constrained processing capacity. These requirements limit the complexity of deployable models, creating a mismatch between the expressiveness needed for robust generalization and the constraints of real-time execution.

Feasible deployment therefore depends not on hardware scaling alone, but on principled model dimensioning aligned with worst-case end-to-end execution timelines and platform characteristics. AI systems intended to replace latency-critical L1–L2 functionality must be engineered explicitly for deterministic real-time operation.

Within this framework, model distillation provides a systematic mechanism to decouple model expressiveness (capacity at training time) from model inference feasibility at deployment time. By transferring generalization capabilities from high-capacity models into compact, RAN-compliant implementations, distillation enables robust performance under stringent real-time constraints while reducing reliance on general-purpose AI accelerators for inference.

References

1. IEEE Communications Magazine, vol. 63, no. 1, pp. 84–91, Design principles for model generalization and scalable AI integration in radio access networks, 2025, Soldati, P.; Ghadimi, E.; Demirel, B.; Wang, Y.; Gaigalas, R.; Sintorn, M. ↗
2. 3rd Generation Partnership Project (3GPP), 3GPP TS 38.211: Physical channels and modulation, 2025 ↗
3. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), New York, NY, USA, Model compression, 2006, Bucilă, C.; Caruana, R.; Niculescu-Mizil, A. ↗
4. Cornell University, Distilling the knowledge in a neural network, 2015, Hinton, G.; Vinyals, O.; Dean, J. ↗
5. International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, Policy distillation, 2016, Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; Hadsell, R. ↗
6. Cornell University, Collaborative deep reinforcement learning, February 19, 2017, Lin, K.; Wang, S.; Zhou, J. ↗
7. International Joint Conference on Neural Networks (IJCNN), Yokohama, Japan, Online policy distillation with decision-attention, 2024, Yu, X.; Yang, C.; Yu, C.; Huang, L.; An, Z.; Xu, Y. ↗
8. International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, Actor-mimic: Deep multitask and transfer reinforcement learning, 2016, Parisotto, E.; Ba, J. L.; Salakhutdinov, R. ↗
9. International Conference on Neural Information Processing Systems (NIPS), Long Beach, California, USA, Distral: Robust multitask reinforcement learning, 2017, Teh, Y. W.; Bapst, V.; Czarnecki, W. M.; Quan, J. ↗
10. International Conference on Artificial Intelligence and Statistics (PMLR), Distilling policy distillation, 2019, Czarnecki, W. M.; Pascanu, R.; Osindero, S.; Jayakumar, S.; Swirszcz, G.; Jaderberg, M. ↗
11. IEEE Conference on Games (CoG), Lisbon, Portugal, Bootstrap your own teacher: Online policy distillation for multi-game reinforcement learning, 2025, Byrne, D. J.; Tot, M.; Duckworth, P.; Bonnet, C.; Laterre, A.; Barrett, T. D. ↗
12. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Practical policy distillation for reinforcement learning in radio access networks, 2025, Khosravi, S.; Demirel, B.; Zhou, L.; Rasines, J.; Soldati, P. ↗
13. IEEE Transactions on Machine Learning in Communications and Networking, Generalization in reinforcement learning for radio access networks, January 19, 2025, Demirel, B.; Wang, Y.; Tatino, C.; Soldati, P. ↗

Further reading

- 5G RAN explained ↗
- AI RAN – Smarter connectivity ↗
- How can the integration of AI in RAN help CSPs enhance network performance? ↗
- Approaching AI-native RANs through generalization and scalability of learning ↗
- Ericsson and Bell Canada successfully test AI-native link adaptation to boost network speed and efficiency ↗

The authors

Pablo Soldati joined Ericsson in 2018 and is currently a principal researcher for AI in radio networks. His research interests include AI, optimization theory and wireless networks. Soldati holds a Ph.D. in telecommunications from KTH Royal Institute of Technology in Stockholm, Sweden.

Cristian Tatino joined Ericsson in 2021 and is currently a concept researcher whose work focuses on AI for RAN automation. His research interests include AI, optimization theory and wireless networks. Tatino holds a Ph. D. in telecommunications from Linköping University in Sweden.

Burak Demirel joined Ericsson in 2020 and is currently a master researcher for AI in RAN. His work focuses on AI, RL, control theory and cyber-physical systems. Demirel holds a Ph.D. in automatic control from KTH Royal Institute of Technology.

Jacek Wszolek bridges industry and academia as a senior specialist at Ericsson and an assistant professor at the AGH University of Krakow in Poland. Specializing in RAN compute machine learning architecture, he focuses on advancing machine learning (ML) capabilities within RANs. Since joining Ericsson in 2019, his work has centered on the intersection of ML and broadband wireless communications. Wszolek holds a Ph.D. in telecommunications from AGH University in Kraków, Poland.

Chakri Padala joined Ericsson in 2007 and is part of the GFTL AIII (AI Innovation and Incubation) team in Bangalore, India. He specializes in building hardware-near software and infrastructure with a current focus on AI hardware. Padala holds an M.S. in computer science from the University of Louisiana, USA.