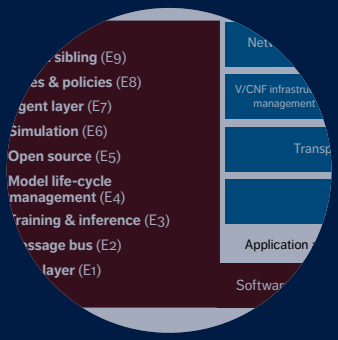
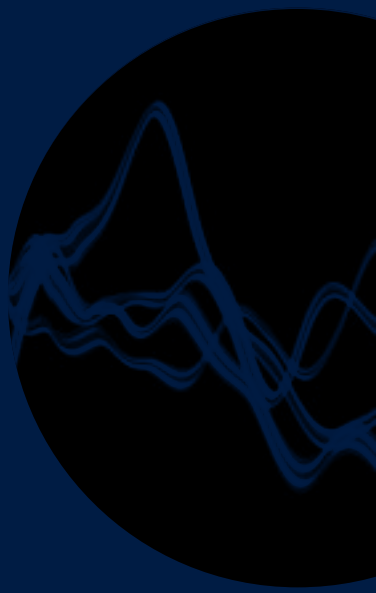
CHARTING THE FUTURE OF INNOVATION | #12·2020

ERICSSON TECHNOLOGY Review





ARTIFICIAL INTELLIGENCE IN RAN

ERICSSON

Artificial intelligence in RAN

A SOFTWARE FRAMEWORK FOR AI-DRIVEN RAN AUTOMATION

Artificial intelligence and its subfield machine learning offer well-established techniques for solving historically difficult multi-parameterization problems. Used correctly, these techniques have tremendous potential to overcome complex cross-domain automation challenges in radio networks. Our ongoing research reveals that an integrated framework of software enablers will be essential to success.

DIARMUID CORCORAN, ANDREAS ERMEDAHL, CATRIN GRANBOM

Modern telecommunications and mobile networks are becoming increasingly complex from a resource management perspective, with diverse combinations of software and infrastructure elements that need to be configured and tuned for efficient operation with high QoS.

■ The latest 5G mobile system is a good example of a sophisticated radio network that allows many deployment variations – such as centralized, distributed or various hybrids of both – while simultaneously supporting diverse categories of applications such as mission-critical control with ultra-reliability and low latency, massively concurrent Internet of Things device access and enhanced mobile broadband.

It is well accepted in the communications community that appropriately dimensioned, efficient and reliable configurations of systems like 5G are a complex technical challenge. Increased real-time, closed-loop, automation at all levels and time frames is a critical tool in controlling this complexity and ultimately reducing the capex and opex of radio network operations.

There is currently a significant trend toward the use of artificial intelligence (AI) technology as an enabler for automating both repetitive and complex tasks. Applied correctly, AI and machine learning (ML) techniques have the potential to boost system

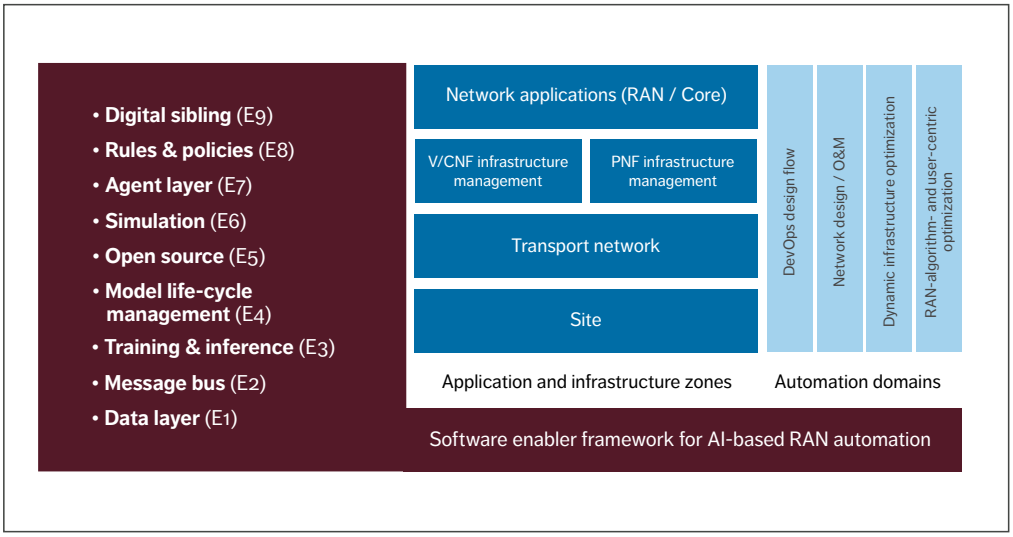


Figure 1 Software-enabler framework for AI-based RAN automation

efficiency by drastically simplifying certain types of automated management and control. It is important to realize, however, that this development will not occur without effort; on the contrary, the need for AI and ML techniques will have a profound effect on how the next generation of RAN should be designed and realized.

As part of this work, we have created a set of software enablers that target efficient and effective use of AI technologies in RANs. Together, they form an important framework of building blocks in realizing the next level of autonomous, AI-based, resource management in 5G and future-generation networks.

Landscape of automation in RAN

The wide variety of architectural and transport options that are possible in 5G results in several different configuration permutations [1]. RAN application(s) can be distributed in a flexible way over both physical (PNF) and virtualized/container (V/CNF) infrastructure, as shown in the middle section of *Figure 1*. How RAN application parameters are chosen and tuned is partially dependent on the selected distribution over compute, transport and site resources. The right side of Figure 1 shows the four automation domains that play a role in optimal network tuning across applications and infrastructure:

- DevOps (development and operations) design flow
- Network design/operations and maintenance (O&M)
- Dynamic infrastructure optimization
- RAN-algorithm and user-centric optimization.

Sophisticated use of AI techniques involves deep and disruptive changes to many existing software engineering, algorithm design and workflow stages. The DevOps (Development and Operations) flow, for example, needs to be extended with new data and ML model life-cycle management (MLCM) procedures. Network design/O&M, dynamic infrastructure optimization and RAN-algorithm/user optimization also need to be reengineered in a radically but carefully staged approach [2].

Software-enabler framework for AI-based RAN automation

Our research and supporting proof-of-concept systems have shown that a set of software enablers is required to best facilitate new AI approaches to sophisticated closed-loop automated network design. These enablers must be tightly integrated into a software framework that allows flexible and open information exchange. As shown on the left side of Figure 1, our software-enabler framework for AI-based RAN automation consists of nine key enablers:

- A data layer (E1)
- A message bus (E2)
- Training and inference (E3)
- Model life-cycle management (E4)
- Open-source AI software (E5)
- Integrated simulation (E6)
- A decision-making agent layer (E7)
- Rules and policies for intent-based management (E8)
- A digital sibling for knowledge harvesting (E9).

Perhaps the most essential enabler of all is the production of industry-standard, machine-readable data (E1) at a high temporal rate by RAN network nodes (eNodeB or gNodeB). This data can be aggregated, stored, filtered and distributed by advanced real-time, hierarchical, message bus (E2) solutions. A distributed training and inference (E3) architecture can efficiently consume this data and use it to facilitate the training of ML models to create new AI algorithms and/or parameter selection for classic algorithm design [2].

There are many alternatives [3] to creating/training models for use with ML algorithms, with three main classifications:

1. Supervised learning, in which algorithms are trained using labeled examples
2. Unsupervised learning, using a cluster or grouping technique
3. Reinforcement learning (RL) [4], using a target reward strategy that allows guidance toward an optimal set of actions.

Depending on the problem targeted, the best ML model and supporting life cycle (E4) can vary considerably in complexity. For simpler problems, linear regression, smaller decision-tree and simple neural networks (NNs) with a few nodes and layers can be sufficient. For more complex problems, large decision trees or deep neural networks (DNNs) with many layers and nodes and several convolutional layers could be necessary to achieve the required accuracy.

RL approaches and supporting agents (E7) for controlling optimization goals can be especially effective at learning novel RAN management strategies. Training RL models depends on active exploration, through a software agent's trial and error experiments, which will not always be possible or appropriate in a live RAN system. To help solve this problem, and to generate the required quantities of data to train models, we have included simulation (E6) in our set of software enablers.

Once trained, an ML model can be used in the inference phase (part of E3), where a selection of data is used as input into the model that will then produce a set of predictions, actions or rules, with the exact details depending on the ML algorithm type. In RAN, the hardware and software requirements on the training and inference phases can be vastly different. Training typically requires powerful central processing unit or specialized graphics processing unit (GPU) hardware with large memory and data storage.

AI software platforms, such as TensorFlow, Keras and PyTorch, together with other extensive, open source (E5), often Python-based, ML software ecosystems need to be integrated into the software engineering flow. During the inference phase, a trained model (or models) is made available to the RAN application(s) through model life-cycle management (E4). For latency-critical RAN edge applications, the inference needs to be efficiently realized, with low latency, power consumption and memory footprint, taking the characteristics of the target hardware and software architecture into account.

Our software enablers are fully compatible with intent-based management solutions [5].

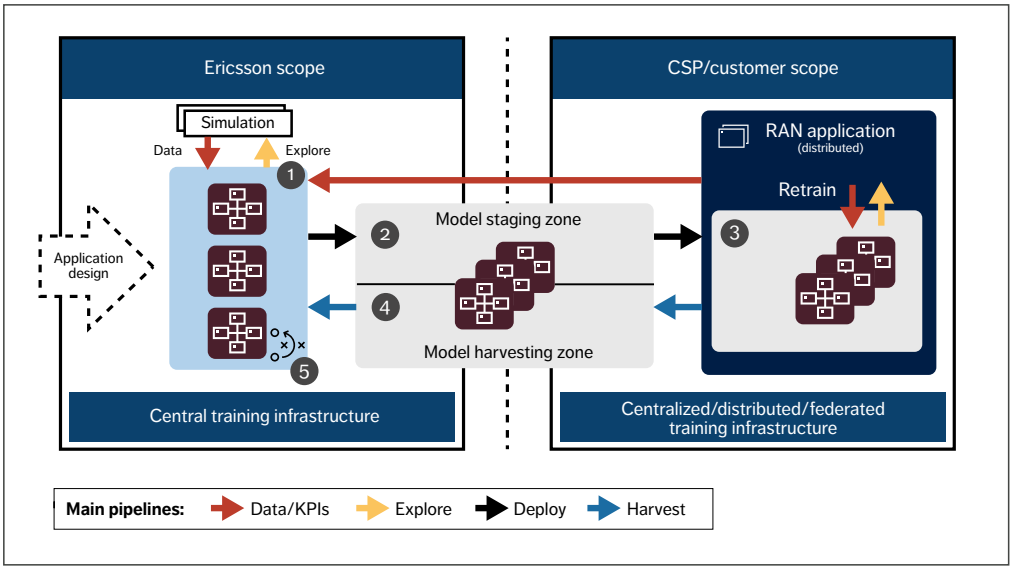


Figure 2 Model life cycle management

When moving toward AI-based automation, operators will no longer spend time tweaking RAN performance by adjusting individual parameters exposed by the RAN system. Instead, well-defined high-level rules and policies (E8) for expressing overall operational intent are used. The AI-based automation machinery is then responsible for realizing these goals.

Long term, to allow the most sophisticated forms of automation, we see a need to build partial digital representations of deployed systems. Our unique digital sibling (E9) concept makes it possible to harvest ML models to build context-specific knowledge.

Model life-cycle management

Machine-learning models are valuable, evolving assets. As such, a sophisticated and controlled software engineering chain (E4) is required for MLCM. When a model is updated through retraining outside its initial context, for example, many significant challenges are introduced in terms of traceability, stability and the possibility to identify faults. With potentially many instances of a model deployed at cell, cell cluster or indeed user level, this challenge is compounded. *Figure 2* captures the main conceptual stages in model management from a software process perspective.

After an initial model design and training phase (marked as 1 in Figure 2), possibly supported by simulation components, the model enters a staging zone (phase 2). This is where customer- or hardware-specific model adaptions [6] occur and are packaged into an industry-deployment format. For example, a DNN can be pruned to minimize nodes that are not significant with additional quantization transformation to reduce the number of bits required to represent a number. This can significantly reduce size, producing an embedded DNN device representation with just a minor impact on accuracy. In phase 3, the model(s) is deployed into a RAN application for use. Initially, these models will be tightly integrated with the RAN application. In the long term, however, it will be necessary to separate model and application life cycles to allow for rapid model training/retraining and deployment cycles.

> WE ENVISION THE USE OF OUR NOVEL MODEL-HARVESTING TECHNIQUE TO CREATE DIGITAL SIBLINGS

Retraining and model drift

Certain types of models (those dependent on traffic and variation in load, for example) may not initially perform well in their deployed context. Over time, there may also be drift between the model training context and the current operating state. As such, it is necessary to allow for periodic, active retraining with context-specific data and features in phase 3.

Model harvesting

Model harvesting is a way to improve simulation and ML training capabilities by enabling a model feedback loop from deployment to design contexts. Phase 4 includes the anonymized collection of model data from a CSP deployment back into our model harvest zone. This facilitates sophisticated data mining from these models, including valuable clustering information on RAN resource usage. The harvested data can then be used as input for model training and retuning when the simulator configurations are updated, for example. In the longer term we envision the use of our novel model-harvesting technique to create digital siblings (phase 5) that partially reflect different real-world RAN deployments.

Test, verification and run-time monitoring of RAN applications utilizing machine learning

A data-driven and continuously learning RAN will introduce new requirements on how software is tested, verified and monitored during system run time. Today, exhaustive test and verification is performed, both on individual software modules and at system level, automated over various continuous integration loops before deployment. The same software is deployed on most RAN nodes, perhaps somewhat differentiated depending on the hardware, surrounding equipment and customer preferences. However, when moving toward RAN systems that utilize ML and RL technologies, this will no longer hold.

For example, ML models may develop independently and not be the same for all RAN nodes, and they may not remain constant between RAN software releases. As a result, new ways to continuously test and validate RAN software utilizing ML models must be developed to fit the MLCM phases.

Moreover, new fine-grained KPIs and metrics must be collected during system run time and fed back to the MLCM. This includes new KPIs for deployed ML models, such as their perceived accuracy, as well as performance and resource consumption KPIs for the RAN applications utilizing the ML models. Similar to other type of RAN data relevant for machine learning, the KPIs should be distributed by the data layer.

A data layer targeting machine learning in the RAN

A cornerstone for producing high-quality ML and RL models is good access to data. Currently, there are many mechanisms for the generation, collection and analysis of data from the RAN. One example is the performance management and configuration management data reported in a standardized format from RAN nodes. Another example is debug and trace data, which includes detailed information about the internal RAN system state. The latter data is often generated on demand – that is, associated with a trouble report or customer service requests.

When targeting generation and use of ML models for time-critical RAN control loops, none of the existing mechanisms is a perfect match. At Ericsson, we are actively investigating and introducing new mechanisms for the generation, management and distribution of data suitable for ML in the RAN. Our lightweight data generation mechanisms allow high quantities of data to be generated without adversely affecting core RAN system performance.

The data is produced using a few standardized machine-interpretable formats. Each data format has an associated schema allowing for easy serializing, deserializing and content interpretation. The data

management allows different data sources to be controlled and activated in a very fine-granular manner. Information on the data and associated schemas is stored and organized in distributed databases, called data- and schema inventories.

Our data distribution is based on message bus technologies, allowing the produced data to be automatically tagged with relevant keys and to be automatically routed to different subscribers based on their tag subscription requests. Our research-developed message bus is hierarchical and distributed, meaning that the produced data will not be routed longer than needed, making it particularly suitable for the requirements of a virtualized and disaggregated RAN.

Enhancing machine learning in the RAN with simulation

Despite ongoing work to improve high-fidelity data collection in the RAN, its distributed nature means it is likely that data will continue to be sparse and that data collection will continue to be a resource-consuming task. At the same time, learning through exploration can be difficult in live systems, where exploration may degrade or crash the system.

Our approach to mitigating these challenges is to integrate in-loop simulation (E6) into our framework, providing the means for much richer data generation as well as the use of explorative RL techniques. Allowing in-loop simulation creates new possibilities for complex, multistep, scenario exploration without compromising system integrity. The first phase of our in-loop simulation is restricted to the Ericsson development scope shown in Figure 2, but it will evolve to include customer deployment in the future.

Looking ahead, more sophisticated AI-based automation in the RAN will require partial digital representations of deployed systems to support complex decision-making. In our framework, we call this enabler a digital sibling (E9).

Digital sibling

The digital sibling is a way to capture and transfer data, and especially ML model data, from a real-world context. These digital representations can then be used for increasingly accurate data generation and in-loop simulation. A related concept, the digital twin, endeavors to duplicate, more completely, a full digital copy of a physical resource or system. The scope of our digital sibling is limited to the transferable digital knowledge, as captured in models that have been trained, interchangeably, in both a simulated and real-world RAN environment.

After an initial training phase, an ML model captures a considerable amount of transferable, problem-specific knowledge, including knowledge that is compressed and coded in the weights of an NN or a set of rules in a decision tree. Sets of models can be used to capture the different decision-making needs of a system. For example, a set of ML models can be trained in a local CSP context to predict the best link adaption or power control parameters for a user.

Other examples include the choice of geographical baseband workload placement required for a given service category and anomaly detection like rogue users or a base station with malicious intent. This aggregated model information (transferred through our MLCM in phase 4), together with actions by goal-oriented software agents, forms a considerable body of knowledge that

Terms and abbreviations

AI – Artificial Intelligence | **CSP** – Communication Service Provider | **DevOps** – Development and Operations | **DNN** – Deep Neural Network | **DSP** – Digital Signal Processor | **GPU** – Graphics Processing Unit | **ML** – Machine Learning | **MLCM** – Model Life-Cycle Management | **NN** – Neural Network | **PNF** – Physical Network Function | **RL** – Reinforcement Learning | **V/CNF** – Virtualized/Container Network Function

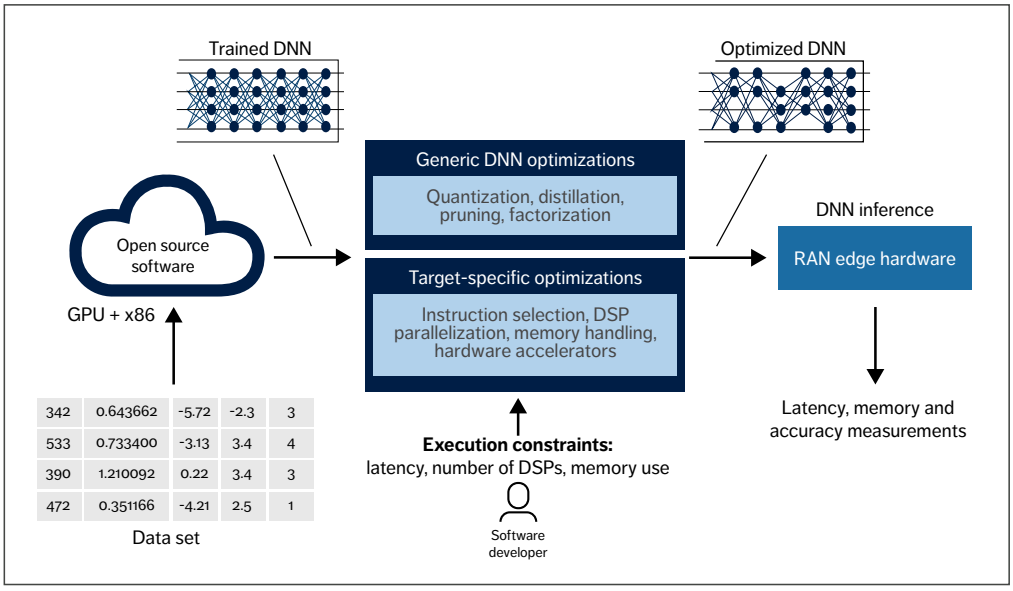


Figure 3 DNN inference optimization for RAN edge hardware

can continuously evolve to form a learning system. The digital sibling is a unique part of our framework, which will ultimately enable advanced automation.

Proof of concept #1 – deep neural network inference optimization

In the proof of concept shown in *Figure 3*, we have investigated how to tune, optimize and realize pretrained DNNs for inference on RAN edge hardware. We selected this example because radio- and baseband data processing are time-critical and computationally intensive tasks. These steps would be part of phase 2 in Figure 2.

The DNN training was done in a container-based environment (Kubernetes) using GPUs and state-of-the-art open source DNN training frameworks. Several generic DNN optimization techniques were investigated and implemented, such as quantization, distillation and pruning, with the goal to reduce the footprint and execution time of the DNN inference, still keeping the accuracy close to the accuracy of the originally trained DNN.

The optimized DNN was realized in terms of digital signal processor-C (DSP-C) code – that is, C extended with some specific DSP keywords and instructions, taking into account constraints given by the target RAN hardware and the RAN software designers, such as the number of DSPs and the amount of memory available for the DNN inference. Latency and memory measurements were made on the RAN edge hardware platform for the generated DNN DSP-C code. We also tuned how to best parallelize the DNN inference over several DSPs.

Based on performed evaluations, we concluded that DNN inference for the target hardware can benefit significantly from the generic- and target-specific DNN optimizations implemented. However, there is normally a trade-off between ML model accuracy, execution time and memory consumption.

Proof of concept #2 – AI-accelerated RAN platform

The proof-of-concept platform shown in *Figure 4*

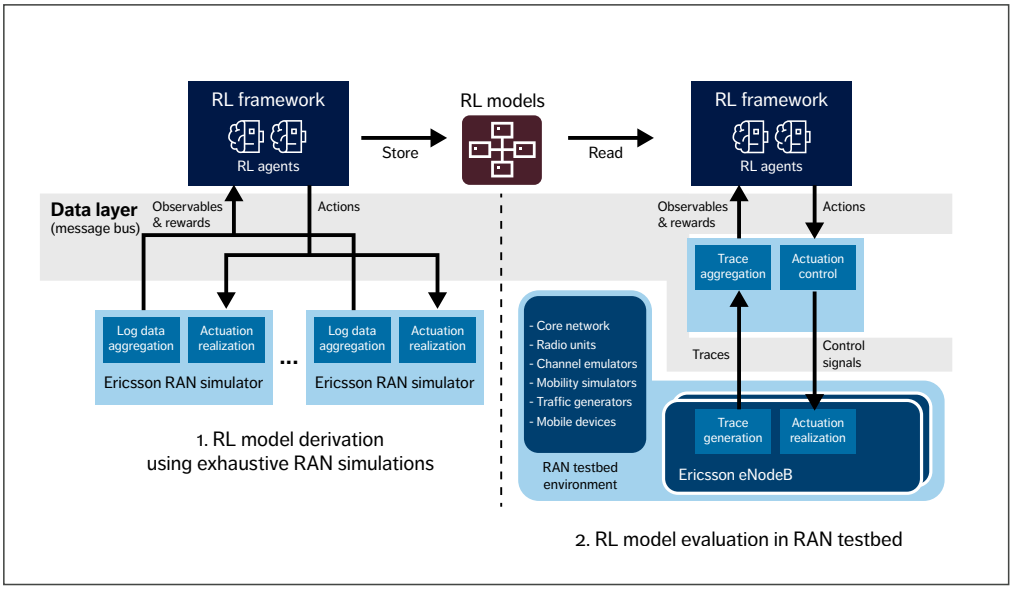


Figure 4 AI-accelerated RAN platform

supports the use of RL techniques and model transfer to control parts of the RAN functionality. Our second proof of concept explores phases 1,2 and 3 in Figure 2. We have used sophisticated RAN simulators to set up and run many different deployment scenarios to train the RL models. The RAN test bed we used consists of several cells and multiple UEs (user equipment). The data layer uses a message bus, which organizes, aggregates and filters data.

To obtain a high degree of scalability and improve model generalization and RL exploration possibilities, the RAN simulators have been containerized, allowing many concurrent instances to execute with different configurations.

The RAN simulator can extract observables and rewards based on internal simulator data, as well as change its internal state and resource allocation strategies based on the actions provided by the RL agents. The RL agent framework is separated from both simulated and actual RAN application by the same well-defined data and action interfaces. RL algorithms can be developed and evaluated in isolation from the RAN application, using RL domain experts, and with required compute and memory resources, including the use of powerful GPUs to train the DNNs that are used by the RL agents.

After achieving high performance in the different simulator scenarios, the simulator-trained RL models can be used for automation use cases in real-life RAN contexts.

Conclusion

Growing complexity and the need to solve repetitive tasks in 5G and future radio systems necessitate new automation solutions that take advantage of state-of-the-art artificial intelligence and machine learning techniques that boost system efficiency. At Ericsson, we have identified the key software enablers for AI- based RAN automation and integrated them into a comprehensive framework that provides a solid and flexible technological foundation for the development of AI-based RAN. Our two proof-of-concept examples, executed in real RAN contexts, clearly illustrate the benefits of our framework of software enablers in building the next generation of automated RAN systems.

References

1. **Ericsson Technology Review, 5G New Radio RAN and transport choices that minimize TCO, November 7, 2019, Eriksson, A-C; Forsman, M; Ronkainen, H; Willars, P; Östberg, C., available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-nr-ran-and-transport-choices-that-minimize-tco*
2. **Ericsson Technology Review, Enhancing RAN performance with AI, January 20, 2020, Calabrese, F.D.; Frank, P; Ghadimi, E; Challita, U; Soldati, P., available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/enhancing-ran-performance-with-ai*
3. **MIT Press, Deep Learning, 2016, Goodfellow, I; Bengio, Y; Courville, A., available at:** *https://www.deeplearningbook.org/*
4. **MIT Press, Reinforcement Learning: An Introduction (second edition), 2018, Sutton, S; Barto, A., available at:** *http://incompleteideas.net/book/RLbook2020.pdf*
5. **Ericsson Technology Review, Cognitive technologies in network and business automation, June 28, 2018, Niemöller, J; Mokrushin, L., available at:** *https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/cognitive-technologies-in-network-and-business-automation*
6. **KTH Royal Institute of Technology, Squeezing and Accelerating Neural Networks on Resource Constrained Hardware for Real Time Inference, 2018, Presutto, M., available at:** *http://www.diva-portal.org/smash/get/diva2:1299467/FULLTEXT02.pdf*

Further reading

- **Ericsson, How will AI enable the switch to 5G?, available at:** *https://www.ericsson.com/en/networks/offerings/network-services/ai-report*
- **Ericsson, Autonomous networks, available at:** *https://www.ericsson.com/en/future-technologies/autonomous-networks*
- **Ericsson, Ericsson launches unique AI functionality to boost radio access networks, available at:** *https://www.ericsson.com/en/news/2019/10/ericsson-ai-to-boost-ran*
- **Ericsson, AI by Design, available at:** *https://www.ericsson.com/en/ai-and-automation*
- **DeepMind blog, AlphaGo Zero: Starting from scratch, available at:** *https://deepmind.com/blog/article/alphago-zero-starting-scratch*

THE AUTHORS

Diarmuid Corcoran

◆ joined Ericsson in 1992. He currently works within the Networks division of Ericsson as an expert in software architecture, actively driving and participating in software-related activities across the company. Corcoran has contributed important software and system solutions to five generations of mobile technology, including 5G. His current research interests include methods, architectures and algorithms for enabling self-learning resource management solutions with a focus on advanced automation. Corcoran holds a B.Eng. in computer engineering from the University of Limerick, Ireland.

Andreas Ermedahl

◆ joined Ericsson in 2010. He works as a software researcher within the Networks division, where he is responsible for driving a software program research area targeting AI and ML in RAN. His research interests include static program analysis, optimization methods and software techniques for facilitating ML and AI in RAN. Ermedahl received a Ph.D. in computer systems from Uppsala University in Sweden and earned the title of docent in computer science at Mälardalen University in Sweden. He currently holds an adjunct professorship in computer science at KTH Royal Institute of Technology in Stockholm.

Catrin Granbom

◆ has worked within Ericsson's telecom and information technology area since 1987, most recently with 4G and 5G. In recent years, her focus has been on research close to implementation, including collaboration ties to academia supporting this. She is currently responsible for software technology research activities within the Networks division, including software implementation technology in areas such as cloud RAN, AI software in RAN, test and quality, high-performance software and software for cyber-security. Granbom holds an M.S. in computer science from Uppsala University.

ISSN 0014-0171
284 23-3351 | Uen

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