From Design to Deployment of Zero Touch Deep Reinforcement Learning WLANs

Ovidiu Iacoboaiea, Jonatan Krolikowski, Zied Ben Houidi, and Dario Rossi

The authors outline successful training and deployment of deep reinforcement learning agents in real-world scenarios facing a number of challenges.

**Abstract**

Machine learning is increasingly used to automate networking tasks, in a paradigm known as zero touch network and service management (ZSM). In particular, deep reinforcement learning (DRL) techniques have recently gained much attention for their ability to learn taking complex decisions in different fields. In the ZSM context, DRL is an appealing candidate for tasks such as dynamic resource allocation, which are generally formulated as hard optimization problems. At the same time, successful training and deployment of DRL agents in real-world scenarios face a number of challenges that we outline and address in this article. Tackling the case of wireless local area network radio resource management, we report guidelines that extend to other use cases and more general contexts.

**Introduction**

The wireless local area network (WLAN) has become the ubiquitous access technology at home, in public locations such as train stations, and in private ones such as university or corporate campuses. Especially in densely populated areas, scarcity of radio resources can easily lead to congestion and thus bad user experience. Luckily, the fleets of WLAN access points (APs) in campus networks can be centrally controlled, opening the way for dynamic and autonomous configuration of network resources: as many of such dynamic resource allocation problems are hard, they are solved in practice using well-thought-out heuristics.

Inspired by the success of machine learning (ML), the field of communication networks has been actively seeking to exploit such techniques to automate complex network tasks, paving the way toward the realization of zero touch network and service management. In particular, deep reinforcement learning (DRL) techniques, which learn by interacting with an environment, are able to achieve complex tasks with unprecedented skills. Top stories include Google’s AlphaGo [1] beating the Go world champion Lee Sedol in 2016, OpenAI Five [2] winning an online computer game DOTA2 tournament in 2017, and recent advances in fully autonomous cars from Tesla [3]. Following a similar path, recent attempts to use DRL instead of heuristics for automating network resource allocation [4], routing [5], WLAN configuration [6], and more [7–9] have shown promising results. At the same time, we observe that while it is relatively straightforward to design and train DRL agents that work well in synthetic and controlled settings, real-world deployment of the same DRL agents poses a set of additional challenges. Indeed, performance evaluation in simplified settings remains a necessary task (i.e., if a solution does not work in simulation, it will never work in the real world), but it is clearly not sufficient (i.e., there are no guarantees that the DRL solution will work as expected in a different environment than the ones on which it was trained). Thus, in order to carry DRL all the way from design to deployment, a number of practical and often underestimated challenges must be accounted for. The latter are just as important as the ML algorithmic design.

Such challenges are rooted in the architectural requirements that must be fulfilled in order for ML techniques to be seamlessly applied in the network. These are nailed down by standardization bodies, such as European Telecommunications Standards Institute (ETSI) Zero-touch Network and Service Management (ZSM), that provide normative architectural references for several tasks. Clearly, as ML techniques are data-driven, a set of requirements concern access to telemetry data, notably the ability to stream it (ZSM req. #84), enforce access control (#86), and store it in data lakes (#87). In particular, DRL model training requires access to data lakes (and likely GPU resources), whereas DRL model inference requires access to stream telemetry (and significantly less computational resources). Furthermore, ML operation requires the ability to deploy and upgrade trained ML models (#46, #49); while some “default” trained model may be necessary for generic zero touch operation, the same model may be “fine-tuned” to the specifics of the environment after deployment. Training a generic model requires historical data gathered from several networks and available in a data lake, while model upgrading requires fresh telemetry for the purpose of fine-tuning. Finally, and most importantly, closed-loop techniques such as DRL need the ability to enforce actions automatically (##68, #115), depending on specific conditions determined by the algorithm, in order to adapt resource allocation better to the instantaneous or forecast evolution of service load.

This article reports our experience in designing and deploying DRL for zero touch WLAN networks. We build on our original design of a DRL sequence-to-sequence architecture, which we previously limitedly validated against state-of-
the-art approaches in simulated settings [6] and have now been running for months on real operational deployments. In the path from design to deployment, we outline and tackle five important challenges related to
1. Safety
2. Duration
3. Realism of the training process
4. Generalization capabilities
5. The adoption barrier of trained models

In sharing our experience with the community, we not only illustrate the specific way in which we overcome such challenges in the WLAN case, but further adopt a broader viewpoint: we complement lessons learned with those gathered from other fields where DRL has been successful, such as gaming [1, 2] and self-driving cars [3, 10], testifying to the generality of these challenges.

The rest of the article presents a high-level view of the zero touch WLAN resource management problem, articulates the main challenges from design to deployment, and summarizes the main lessons.

**Zero Touch Deep Reinforcement WLANs**

Our goal is to autonomously manage WLANs in closed loop, continuously adapting allocated radio resources to changing traffic conditions and demand to maximize end-to-end performance. We first briefly cover WLAN management, which we next reconsider under the lens of zero touch closed-loop control, and finally overview the DRL technique we employ.

**Wireless LANs**

WLANs are defined in the IEEE 802.11 standard and its amendments. Zero touch operation in the general case [8] and in heterogeneous, industrial, and enterprise WLANs are surveyed in [9]. Here, we provide a very basic overview of WLAN resources and actions from the viewpoint of autonomous closed-loop control and refer the reader to [11] for a recent survey on ML applications to WLANs. The most popular WLAN setup is infrastructure-based, where stations (e.g., smartphones, laptops, or industrial devices, referred to as STAs) connect to fixed access points (APs) that typically act as gateways to relay STA traffic. In office buildings or university campuses, a fleet of APs is deployed over a large area to connect the numerous STAs to the Internet. Typically, central management decisions are made to optimally manage the network: the set of actions includes, for each AP, selecting a channel, bonding (half-duplex manner), an AP may be configured to allow the aggregation (a.k.a. bonding) of several channels to increase bandwidth and consequently throughput. Ideally, only one device within the receiver’s vicinity transmits on one channel at the same time, avoiding collisions and data loss. This is achieved through the listen-before-talk mechanism of carrier sense multiple access with collision avoidance (CSMA/CA).

As a consequence, APs and STAs on the same channel share airtime: the time that a transmitter waits while the channel is busy is called interference (time). Depending on the regulatory region, only 4 (20) channels are non-overlapping on the 2.4 GHz (5 GHz) band: thus, it is not always possible to allocate different channels to neighboring APs, and in highly dense areas, interference cannot be avoided by simple channel allocation. It follows that on top of channel allocation and bonding [12], the AP transmit power [13, 14] can be additionally used to trade off the strength and quality of the received signal vs. the airtime interference. While in a number of scenarios (e.g., household AP configuration decisions would benefit from being delegated to the APs, in this work we limitedly consider the case of centralized radio resource management for the sake of simplicity.

**Zero Touch WLANs**

We note that the network configuration has to be selected from a very large state space that grows exponentially with the set of available configuration knobs. Additionally, as the network load evolves over time, it would be desirable for the WLAN to be able to autonomously adjust its configuration to best adapt the available resource to the current (or forecast future) demand. As the utility function to estimate network quality can be a complex combination of quality of service (QoS) (e.g., signal strength, coverage, interference, user throughput, latency) and quality of experience (QoE) indicators (e.g., more advanced per-application metrics), autonomous configuration becomes a desirable capability of WLAN networks. From this viewpoint, with reference to Fig. 1, it can be envisioned that a zero touch WLAN is governed by an ML model making decisions (e.g., configuration actions) as a function of the observable state (e.g., stream telemetry). Such a model should be pretrained (e.g., by using a digital replica of the network), but could possibly benefit from specific finetuning from real data after deployment (to upgrade the model in the long run).

We observe that the existence of two separate environments results in a dichotomy inducing three separate loops: a first design/train/validation cycle (Fig. 1, left) where the ML model is trained on a digital replica of the network; a second deploy/inference/test cycle (Fig. 1, right) where the trained model is used on the actual network; and a third refinement loop, bridging the two environments. The picture also highlights
several important practical aspects that this article is going to dissect, notably: pre-training is necessary to train safely and train fast, but note the need for environmental realism for better fit and generalization capabilities to unknown states. Finally, explainability and trust are key to deploy zero touch closed-loop operation.

**Deep Reinforcement WLANs**

DRL techniques are suitable for implementing the closed-loop control algorithm. To better understand challenges that DRL agents may face in real-world deployment, we first briefly remember its most important concepts and cast them to WLAN with the help of Fig. 1. Without loss of generality, in the context of this work we limit the configuration knobs to the selection of primary channel and bonding.

**Background on RL and DRL:** In reinforcement learning (RL) [15], an agent learns from interacting with an environment: the agent obtains a perception of the environment through a measurable state (network configuration, stream telemetry, etc.) and selects an action (e.g., changing an AP configuration) based on a policy that it is learning. After enforcing the action, the agent observes an updated state and receives feedback about its action in the form of a reward (or a regret). Unlike in supervised learning (where the feedback reflects some distance from the optimum), the feedback in RL can be seen rather as praise (or critique) of the action (without any explicit information about the optimality). Based on this feedback, the agent updates its policy, observes the new environment state and enforces a new action, learning to increase its reward (or decrease its regret). DRL is a class of approaches that are based on a neural network (NN), where the NN is used to learn either the value of the state (i.e., DQN) or the policy (e.g. A2C, A3C). A recent wave of DRL approaches have shown interesting results in the solution of combinatorial graph problems, using various architectures such as graph NNs, pointer networks, and graph attention networks [7].

**Overview of WLAN DRL Agent:** In our previous work [6], we used a similar philosophy to develop a DRL architecture fit for WLAN channel management. While in [6] we limitedly validated the approach against the state of the art via simulation (Fig. 1, left), in this article we are concerned about the complementary necessary steps for real-world deployment (bridging the gap between the left and right of Fig. 1). As such, we provide here only a necessary limited overview of the WLAN DRL in Fig. 2, and refer the reader to [6] for details.

Our design follows a classic actor-critic NN architecture, where the critic-NN, which learns the value function, guides the actor-NN in learning the best policy, to which we add a selector-NN, guiding the choice of the best action among those returned by multiple parallel runs of the actor-NN. The actor-NN employs an encoder-decoder sequential architecture, where basically the same NN is run sequentially multiple times, making at each step a decision for one of the APs in the network. In particular, our encoder-NN employs a careful feature engineering process to transform variable-size input features (depending on network size, number of channels, etc.) into a fixed-size intermediate representation that is used as input by the decoder-NN to output a probability distribution over all actions (i.e., one channel and bandwidth option for each AP), which makes it suitable for application to arbitrary networks.

**From Design to Deployment**

We now report on our deployment experience of DRL agents in real operational WLANs. In particular, our DRL-based WLAN channel management solution (agent) autonomously reconfigures in closed loop (action) every 10 minutes; hence, the need for accurate forecast of future demand is lessened by the fact that actions are frequently taken in a real operational WLAN (environment) based on telemetry data (state) received in a sub-minute timescale.

While training and deploying this DRL-based system, we faced a series of challenges that were mentioned in Fig. 1, into which we now dig deeper—first summarizing our experience with WLAN deployment, and then contrasting the lessons learned to other DRL real-world use cases.

**Train Safely**

**WLAN Insights:** Training on the operational WLAN network would inevitably lead to the exploration of bad network configurations harming user experience, which is not a viable option. Since an expensive WLAN testbed (where bad network configurations would be permissible) is not available, we are forced to train the DRL agent using a digital replica of the system, i.e., a computer-simulated model of the real environment. While learning from a digital “twin” solves the safety concerns altogether, it does, however, introduce another trade-off. Namely, the simulator needs to be realistic enough to favor the transfer to the real deployment and at the same time simple enough to allow for reasonable training time, which are both key aspects that are worth digging into deeper later.
**Beyond WLAN**: In autonomous driving, the necessity of training safely is even more obvious. For example, AWS deepracer [10], a platform to train and test DRL agents for this use case, relies on a cloud-based 3D racing simulator as one key component that helps avoid the exploration of the most detrimental states even with model cars. Tesla also trains its Autopilot offline before allowing it on the road [3].

**Train Fast**

**WLAN Insights**: Two factors impact the training duration of our agent:

- The convergence of DRL weights during the training process, which affects the number of interactions
- The duration of each simulated interaction, during which the DRL training process remains idle waiting to receive state and regret feedback from the environment

As far as the first, we calibrated the training phase carefully to avoid getting stuck in local minima, mainly by adjusting the learning rate during training such that more aggressive updates (e.g., larger steps) are performed at the beginning, followed by smaller steps allowing the system to gradually stabilize. As for the second, the duration of a simulated interaction is by far the dominant bottleneck, for which we rule out the use of packet-level simulations (e.g., ns-2 or ns-3) and leverage a fast custom simulator, with low computational complexity. Figure 3 illustrates the regret evolution over multiple independent training runs: the x-axis reports the number of iterations, the GPU training time (including the simulation time, measured in hours) and the equivalent duration of the training process had it been performed at the same timescale in a real environment (measured in years).

**Beyond WLAN**: Training duration is a clear bottleneck in any DRL deployment. In the most recent successful DRL applications, agents need several “lifetimes” of interaction with the environment (e.g., the 10,000 years equivalent of gameplay for OpenAI Five [2]), which is clearly unrealistic for training on real systems. Even training offline with real data can take a significant amount of time: for instance, it took 70,000 GPU hours to train the full self-driving Tesla pilot prototype [3], which is around one year for a single node with 8 GPUs. In some cases, training needs to be offloaded to a large fleet of data center servers equipped with GPUs and TPUs, which may only be affordable for a few big players. In network usecases, depending on the size of the DRL NN, the digital “twin” can become the computational bottleneck.

**Environment Realism**

**WLAN Insights**: As environment realism remains a key concern, we can leverage real-world data to enhance the simulator models — bridging the training and validation environments. To retain scalability while enhancing realism, we use a data-driven approach to refine several models used in the simulator. To make just a single example in reason of space limits, our simulator uses a threshold on the received signal strength (RSSI) to decide which APs are considered as neighbors: we increase realism by fitting this parameter to maximize similarity between the interference estimations of the simulator and those measured in the real network. As can be observed in Fig. 4, the ideal RSSI threshold under which two APs should be considered neighbors is at ~82 dBm, with which we calibrate the simulator for fine-tuning the DRL agent.

**Beyond WLAN**: Environment realism is key when an agent trained on simulation is to be transferred into the real world. Beyond WLAN, this is witnessed by the amount of efforts that are put into environment realism. For example, beyond the AWS deepracer 3D simulator [10] we mentioned earlier, various environments such as Learn-to-race and Torcs compete to offer better reinforcement learning conditions. Numerous examples exist in other related industries such as energy, robotics, manufacturing, and supply chain.

Beyond faithful simulation, imitation learning is a valid complementary approach, where instead of interacting with a real or simulated environment, a database of state-action traces is used for offline batch-based agent training. For instance, thanks to its fleet of several hundred thousand self-driving-ready cars, Tesla now has a huge amount of state-action pairs (over 1 billion miles with Autopilot-on [3], which might be used to learn to mimic human driver behavior.)
Generalization

WLAN Insights: Generalization to conditions unseen during training means, on the one hand

- Generalizing to WLAN networks of arbitrary size and density, and on the other hand
- Transferring well to the more complex physics of the real network — which are both necessary as deployment conditions will never match exactly the training conditions.

We tackle the first by a novel auto-regressive sequential decoder whose input features at each step are engineered to reflect the changing internal state of the decoder that is described in [6].

As for the second, robustness of the DRL agent to varying conditions is key: to test the ability of the trained DRL model to cope with unknown conditions, we train the agent in an ideal environment and test it in a noisy one. We systematically apply Gaussian noise with controlled means and standard deviations to the AP neighborhood (i.e., RSSIs each AP sees from all others) and observe the impact on the regret. By choosing Gaussian noise, we do not seek realism but rather understand the model’s ability to generalize to unseen conditions. The left of Fig. 5 reports the relative percentual increase of the regret with respect to ideal conditions (noiseless training and testing) when neighborhood is defined with a simple threshold. Overall, the picture confirms DRL to be robust for a wide range of additive noise.

Additionally, consider that negative noise causes the corresponding RSSI to fall below the neighborhood threshold, leading to interference underestimation. Unsurprisingly, the figure confirms underestimation to be more harmful than overestimation, which validates our conservative simulator design choice.

However, in the real network, neighborhood is not exactly a clear-cut threshold: the right side of Fig. 5 further tests the algorithm on the same noisy conditions, but using a more complex (but still unrealistic) neighborhood definition. In particular, neighborhood interference is smoothly taken into account by using an S-shaped sigmoid function (with a spread of 6 dB centered around the clear-cut threshold). Interestingly, results of DRL trained in ideal settings remain robust even under these different testing environmental conditions, which bodes well for DRL generalization ability in a real environment.

Beyond WLAN: Generalization in DRL is still an open and active research area. One recent way to favor generalization is to leverage contrastive learning to force states with similar behavior to have similar representations. Another way to generalize, closer to ours, is to leverage faithful simulations and fine-tuning in real conditions, as was successfully demonstrated in the context of vision-based flight control.

EXPLAINABILITY AND TRUST

WLAN Insights: Lastly, it is essential that network engineers develop trust in the algorithm decisions before letting it run unattended on thousands of customer deployments. As DRL decisions are intrinsically less interpretable than heuristics, this can first be achieved by human-understandable explanations of the algorithm decisions and expected gains so that the WLAN operator can not only perceive DRL operation as safe, but also understand and value its benefits.

In the context of our research, we conducted months of tests in operational WLAN networks to illustrate the DRL agent’s viability. In particular, we ran several batches of experiments, each lasting one week, in which we either run:

- The trained DRL agent every 10 minutes to closely track load changes vs.
- A daily static optimization based on historical load forecast.

Here, we show an excerpt of deployment results in a campus network in Nanjing, China. Figure 6, left, contrasts the channel utilization on a 30 AP WLAN with several thousand STAs on a typical day. We construct a heatmap from the scatter plot where each point represents the average channel utilization for the same AP during 10 minutes at the same time of day and day of the week for the two algorithms over all APs: this allows us to assess the impact of dynamic DRL channel management from a spatial viewpoint (i.e., from the point of view of the same AP). We can note the tendency of improvements as the center for the highest density moves above the diagonal; the static, indeed, has more APs with higher load compared to DRL.

Clearly, while the traffic is similar every week due to seasonal behavior of the users, the traffic conditions are not identical, which can bias the comparison. Trust in the solution can then be gained only through careful analysis over long-term campaigns. For instance, we take this confounding factor into account by comparing in Fig. 6, right, the breakdown of the AP utilization (y-axis) for the highest den-

sity moves vs. the diagonal: the highest den-

sity moves accordingly.

Beyond WLAN: As a general lesson, human operators need to gain understanding and build trust in DRL systems to allow their deployment. Explainability of the algorithm output helps lower the adoption barrier. Trust may then be gained step by step: convincing results from extended experiments on real-world deployment help show the benefits of the algorithm — which holds for any use case.

For instance, Google DeepMind adopted the following strategy for data center cooling: the first
RL algorithm version acted only as a recommendation engine for the operator. Later, they moved to a fully autonomous version, maintaining a fail-safe option to revert back to human control at any time, in addition to rule-based heuristics as backup.

**Conclusion**

DRL is a promising paradigm for controlling complex systems, improving decision making over human intuition and classic heuristics. While most network-related DRL research focuses on ideal scenarios and evaluated via simulation, we discuss here the challenges that arise when DRL is deployed in a large-scale operational WLAN to achieve zero touch operation.

We generalize and summarize the lessons learned as follows. It appears that DRL training requires digital “twins,” such as simulators. Indeed, solely learning from existing network data may be neither feasible (given the sheer number of samples needed for training) nor desirable (as it does not offer enough action diversity, so missing unsafe actions). Conversely, learning from simulation provides the best trade-off among safety (i.e., to also explore unsafe actions), simplicity (for training duration), and realism (e.g., simulators can be enhanced with real-world data and be used to assess controlled generalization). Finally, technical benefits are a necessary (but insufficient) condition to adoption: deployment of trained DRL models for real-time inference still requires a pedagogic effort toward the human operators interacting with it (in order to gain their trust), as well as offering fallbacks to legacy systems until the algorithm gains sufficient trust for true fully automated zero touch operation.

**References**

[1] D. Silver et al., “Mastering the Game of Go without Human Knowledge,” Nature, vol. 550, no. 7676, 2017, pp. 354–59.
[2] https://openai.com/projects/five/, accessed Jan. 7, 2022.
[3] A. Karpathy, “Pytorch at Tesla,” Pytorch DevCon ’19, 2019.
[4] H. Mao et al., “Resource Management with Deep Reinforcement Learning,” ACM HotSocs, 2016, pp. 50–56.
[5] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, “Learning to Route,” ACM HotSocs, 2017.
[6] O. Iacoboaiea et al., “Real-Time Channel Management in WLANs: Deep Reinforcement Learning versus Heuristics,” JIP Networking, 2021.
[7] N. Veselova et al., “Learning Combinatorial Optimization on Graphs: A Survey with Applications to Networking,” IEEE Access, 2020.
[8] C. Benzaied and T. Taleb, “AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions,” IEEE Network, vol. 34, no. 2, Mar./Apr. 2020, pp. 38–42.
[9] M. Friesen, L. Wisniewski, and J. Jasperneite, “Machine Learning for Zero-Touch Management in Heterogeneous Industrial Networks — A Review,” IEEE Int'l. Conf. Factory Commun. Sys., 2022.
[10] https://aws.amazon.com/deepracer, accessed Jan. 7, 2022.
[11] S. Szott et al., “WiFi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning,” IEEE Commun. Surveys & Tutorials, vol. 24, no. 3, 2022, pp. 1843–93.
[12] A. Bhatia et al., “Measurement-Based, Practical Techniques to Improve 802.11ac Performance,” ACM IMC, 2017.
[13] N. Ahmed and S. Keshav, “A Successive Refinement Approach to Wireless Infrastructure Network Deployment,” IEEE WCNC, 2006.
[14] V. Shravastava et al., “Understanding the Limitations of Transmit Power Control for Indoor WLANs,” ACM IMC, 2007.
[15] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, 2018.

**Biographies**

OVIDIU-CONSTANTIN IACOBOAIEA (ovidiu.iacoboaiea@huawei.com) is a senior research engineer at the DataCom Lab of Huawei’s Paris Research Center. He received his M.Sc. from the University Politehnica of Bucharest, Romania, and Supélec, France, in 2012, and his Ph.D. degree in self-organizing networks from Telecom ParisTech, France, working with Orange Labs in 2015. Afterward, he worked for Bouygues Telecom until 2019 as a system engineer responsible for mobile network analysis, modeling, optimization, and dimensioning. His current interests include network optimization, artificial intelligence, machine learning, and big data.

JONATAN KRZELIKOWSKI (jonatan.krzelikowski@huawei.com) is a senior research engineer at the DataCom Lab of Huawei’s Paris Research Center. He received his M.Sc. in mathematics from TU Berlin in 2014 and his Ph.D. from Université Paris-Sud/Centrale-Supelec/LSI in 2018 on the topic of optimal content management and dimensioning in wireless networks. His current research interests are network optimization applied to real networks leveraging methods from machine learning and operations research, as well as modeling and analysis of network related problems.

ZIED BEN HOUID (zied.ben.houidi@huawei.com) is a principal engineer at the DataCom Lab of Huawei Paris Research Center. He received his Ph.D. from Université Pierre et Marie Curie, France, in 2010 while he was working at Orange Labs on data-driven performance analysis of core networks’ routing protocols. He then joined Bell Labs, the research arm of Nokia. There, he proposed and led various research projects on network data valorization (e.g., human-level behavior analytics, learning from network/ISP data to build recommender systems) as well as automated reasoning for programmatic specifications/standards generation. The projects led to several deployments, patents, and demos as well as publications in top tier competitive venues in networking (e.g., ACM HotNets) and social computing (ACM CSCW/Pyrx). ACM on CHI, ACM Trans. on the Web).

DARIO ROSSI (SM) (dario.rossi@huawei.com) is director of the DataCom Lab at Huawei Paris Research Center. Previously, he was Chair Professor at Telecom ParisTech (2006–2018) and a professor at Ecole Polytechnique (2012–2019). He received his M.Sc. (2001) and Ph.D. (2003) degrees from Politecnico di Torino, his HDR (2010) from Université Pierre et Marie Curie, and was a visiting researcher at the University of California, Berkeley (2003–2004). He has coauthored 15+ papers and 200+ papers on different aspects of networking that attracted 7500+ citations and an H-index of 43. He received 9 best paper awards, a Google Faculty Research Award (2015), and an IRTF Applied Network Research Prize (2016). He is a Senior Member of ACM.

**FIGURE 5. Explainability (left): channel utilization heatmap, comparing same AP and same time-of-day slots across different algorithms (static and DRL) over different days. Trust (right): statistically unbiased comparison of individual AP load (y-axis) for same average network load (x-axis).**

IEEE Communications Magazine • February 2023