Towards Self-Driving Radios: Physical-Layer Control using Deep Reinforcement Learning

Samuel Joseph*
Stanford University
josamuel@stanford.edu

Rakesh Misra*
Stanford University
rakeshmisra@alumni.stanford.edu

Sachin Katti
Stanford University
skatti@cs.stanford.edu

ABSTRACT

Modern radios, such as 5G New Radio, feature a large set of physical-layer control knobs in order to support an increasing number of communication scenarios spanning multiple use cases, device categories and wireless environments. The challenge however is that each scenario requires a different control algorithm to optimally determine how these knobs are adapted to the varying operating conditions. The traditional approach of manually designing different algorithms for different scenarios is increasingly becoming not just difficult to repeat but also suboptimal for new scenarios that previous-generation radios were not designed for.

In this paper, we ask: can we make a radio automatically learn the optimal physical-layer control algorithm for any scenario given only high-level design specifications for the scenario, i.e., can we design a *self-driving radio*? We describe how recent advances in deep reinforcement learning can be applied to train a self-driving radio for several illustrative scenarios, and show that such a learning-based approach not only is easily repeatable but also performs closer to optimal than the current state of the art.

CCS CONCEPTS

• **Networks → Wireless access points, base stations and infrastructure**; *Mobile networks*; • **Theory of computation → Reinforcement learning**.

KEYWORDS

Radios; Physical layer; Cellular networks; Reinforcement learning; Deep learning; LTE; 5G

ACM Reference Format:
Samuel Joseph, Rakesh Misra, and Sachin Katti. 2019. Towards Self-Driving Radios: Physical-Layer Control using Deep Reinforcement Learning. In *The 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19), February 27–28, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301293.3302374

*equal contribution.

HotMobile '19, February 27–28, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6273-3/19/02…$15.00
https://doi.org/10.1145/3301293.3302374

1 INTRODUCTION

Modern radios[1], such as 5G New Radio (NR) [1], are expected to accommodate a wide variety of communication scenarios [6]. They are at once expected to be able to support high-bandwidth fixed wireless access, e.g., to provide fiber-like broadband connectivity to the home, very high-speed mobilities, e.g., trains moving at 300 km/h, and very low-power connectivity for IoT devices, among many others. No single physical-layer design can work well under all scenarios, hence the natural response of the standards bodies has been to specify designs with a large number of control knobs so that a radio can be tuned to the specific deployment scenario in the field. These control knobs range from modulation order and coding rate, to OFDM subcarrier spacing and cyclic prefix length, to reference signal (pilot) density, to transmit power scaling and so on. Each of these knobs has numerous settings leading to a combinatorially large number of choices, and radio designers need to design a control algorithm that chooses the right one dynamically at run time depending on the scenario and the varying operating conditions.

The current approach is to preprogram empirical control rules with configurable thresholds that can be adjusted by domain experts in an iterative trial-and-error manner on the field for each scenario. This approach has worked reasonably well for LTE but is under strain for 5G because of the much larger *number* and *complexity* of scenarios it is expected to support, e.g., LTE was not originally designed to support fixed wireless access or low-power IoT connectivity. Furthermore, 5G is expected to evolve to support network slicing where the wireless stack is adapted to future applications, e.g., peer-to-peer connectivity for autonomous vehicles, that are not fully defined yet, so an approach based on heuristic rules and thresholds is intrinsically limited in *how quickly* and *how optimally* it can be repeated to support new requirements.

In this paper, we argue for a learning-based approach to designing control algorithms for radios. Our vision is that radio designers should be able to provide only high-level specifications for a scenario, including the communication objective, the control knobs and measurements available, and a model for the wireless environment in that scenario, and the radio should be able to learn on its own an algorithm to adapt the control knobs in real time[2]; see Figure 1. In essence, the goal is to *learn*, rather than design, a near-optimal control algorithm that can be deployed on the field. We refer to such radios as *self-driving radios* (SDR). In doing so, we hope to repurpose the SDR acronym, originally used for a software-defined radio, to refer to a radio that has not only the flexibility to be configured in software (software-defined) but also the capability to build the right software to use in any given scenario (self-driving).

[1] In this paper, by *radio*, we refer to the radio baseband, not the front end.
[2] Note: this algorithm does not require access to user data, e.g., I/Q samples.

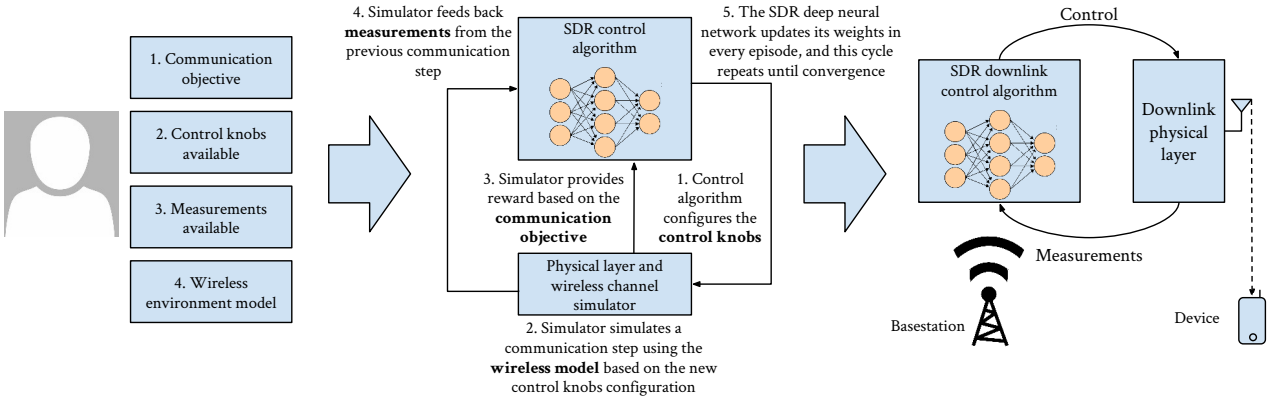

Figure 1: The self-driving approach to radio design. Unlike the traditional approach where radio designers manually design the physical-layer control algorithms, in this approach, they just manually specify the communication scenario and the radio learns a near-optimal control algorithm on its own using deep reinforcement learning.

We describe the SDR approach in more detail in Section 2, and contrast it with related work in Section 3. We illustrate the SDR approach in Section 4 by showing how to design control algorithms to jointly control the *modulation order* and *transmit power scaling* in several scenarios where a radio has to meet a throughput target while minimizing its average transmit power. We show in Section 5 that control algorithms using the SDR approach are able to meet their throughput targets 10-45% more often than the control algorithms in LTE, which is the industry state of the art, and only 7-26% less often than the offline-optimal control algorithms, while using similar transmit powers as LTE and 2-3 dB higher than optimal.

2 MOTIVATION FOR THE SDR APPROACH

In this section, we describe the two broad challenges in designing physical-layer control algorithms for modern radios using traditional approaches, and explain why a self-driving approach has the potential to overcome both challenges.

Challenge: should be quickly repeatable for new scenarios The physical-layer control algorithms for modern radios such as NR need to be easy to adapt to new, potentially unseen, scenarios in the future. These scenarios can be very diverse, and may span multiple use cases, device categories and wireless environments. Different use-case requirements, e.g., high throughput, low latency, high reliability etc., present different optimization objectives. Different device categories, e.g., low-power IoT, high-mobility IoT, fixed wireless etc., mean that the optimization must be performed under different device constraints. Different wireless environments, e.g., indoor vs outdoor, short range vs long range, stationary vs mobile etc., present different dynamics, which require different optimization strategies over time. As a result, every new combination of use case, device category and wireless environment requires a very different control algorithm. The traditional approach, which involves manually designing a control algorithm for every scenario using empirical rules and thresholds, is tedious and inherently limited in how quickly it can be repeated for new scenarios.

SDR approach: learn rather than design The SDR approach aims to use machine learning to *learn* the optimal control algorithm. A learning-based approach is quickly repeatable for new scenarios because designing a new control algorithm just requires manually *specifying* the new scenario, see Figure 1, as opposed to manually *designing* a new algorithm, e.g., by fine tuning rules and thresholds.

Challenge: should sequentially optimize for a communication objective while controlling inter-dependent knobs In order to meet the needs of a wide variety of stressful scenarios, modern radios need to optimize for an explicit objective, e.g., an IoT radio may want to reliably upload a fixed-size packet under a total energy constraint, while controlling several control knobs that we had listed earlier. The challenge is two-fold. *First*, this optimization problem is sequential, i.e., a control decision in one time step affects the optimal decision in a subsequent step, e.g., the above IoT radio may choose to transmit at a higher power in the beginning to increase the reliability of the upload but will be left to work with a lesser energy budget subsequently. In traditional approaches, the decisions in one time step do not explicitly account for the consequences on future steps. *Second*, the set of control knobs is not just large but also inter-dependent, e.g., a drop in link quality can be handled by either reducing the modulation order or increasing the transmit power, among several other ways. The traditional approach uses separate algorithms to control these inter-dependent knobs, e.g., modulation and coding rate is chosen by a separate algorithm than the transmit power. As a result, as the scenarios become more complex, e.g., the wireless environment becomes more dynamic or the set of inter-dependent knobs becomes larger, the traditional approach becomes increasingly suboptimal.

SDR approach: learn by reinforcement using deep neural networks The SDR approach aims to learn the *joint* optimal control algorithm using *deep reinforcement learning* (RL). RL is a natural

fit for sequential optimization and has been successfully applied in a number of problems [7, 9, 10] where a controller has to learn how to plan its actions with future consequences in mind. Deep neural networks (DNNs), owing to their large expressive power, are capable of encoding the complex strategies required for adapting the large inter-dependent set of control knobs. Therefore, a control algorithm expressed using a DNN and trained using RL has the potential to be near optimal, even in the most complex of scenarios.

3 RELATED WORK

Deep learning has been applied extensively in recent times to solve problems arising in mobile and wireless networking; see [15] for an excellent survey. Most of these works however relate to either (i) mobile data analysis, which involves using deep learning to solve *prediction* or *clustering* problems e.g., network prediction [11], traffic classification [12] etc., in contrast to SDR which uses deep learning for a *control* problem, or (ii) mobile network or link control, which involves using deep learning for mobile *network-layer* or *link-layer* control algorithms e.g., traffic scheduling [3], resource allocation [14], interference alignment [5] etc., in contrast to SDR which uses deep learning for *physical-layer* control.

Deep learning for wireless physical-layer control has been relatively much less explored. The closest we are aware of are works that used deep learning for building control algorithms for specific physical-layer knobs, e.g., closed-loop power control for VoLTE downlink [8], adaptive modulation and coding [4] etc. SDR is different from these works in two significant respects. First, SDR learns the *joint* optimal control algorithm, i.e., an algorithm to control all physical-layer knobs jointly for optimal performance, unlike these works that focus on individual control knobs independently. Second, SDR optimizes *explicitly* for the communication objective, unlike these works which focus on optimizing lower-level metrics like SINR which may not lead to the optimal communication performance, e.g., an energy-constrained IoT radio that wants to conserve battery does not care about maximizing SINR.

4 DESIGNING A SELF-DRIVING RADIO

Figure 1 illustrates the key steps in the design and deployment of a self-driving radio. We describe these key steps below.

A radio designer provides design specifications. First, a radio designer provides high-level design specifications for the physical-layer control algorithm, specifically: (i) the objective function that it must optimize, (ii) the control knobs that it must adapt, and their possible settings, (iii) the measurements that it can use to adapt the control knobs, and (iv) a channel model, or a set of channel traces, representing the characteristics of the wireless environment that it must optimize its strategy for.

A learning engine generates a physical-layer control algorithm. Next, a learning engine converts the above specifications into a physical-layer control algorithm, encoded typically as a DNN. This learning engine forms the core of the SDR capability. We will describe it in more detail in the rest of this section.

A radio uses the learned control algorithm at run time. Finally, DNNs that are learned using the above learning engine are used at run time to adapt the physical-layer control knobs for both downlink and uplink. In practice, we envision that any base station, depending on its purpose and deployment, would typically need to support anywhere between 1-3 use cases, 2-4 device categories and 1-5 operating environments, so it might need to maintain on the order of a few such DNNs to tens of such DNNs[3]. At run time, before every transmission, a base station will first select the most relevant DNN for each of uplink and downlink, and then execute them to adapt the control knobs. The decisions of the DNNs will be enforced at the base stations for the downlink, and signaled via the control channel to the devices to enforce for the uplink.

In the rest of this section, we focus on the design of the SDR learning engine. To illustrate its working, we use the following scenarios as reference, in each of which a radio needs to meet an average throughput target in every 10-ms window while optimizing its average transmit power based on its energy sensitivity, and describe how to learn control algorithms for these scenarios.

Scenario 1: A radio on a phone needs to achieve 3-Mbps physical-layer throughput on the uplink in every 10-ms time window, to support, for example, HD (720p) real-time video conferencing.

Scenario 2: A radio on a connected autonomous vehicle (AV) has the same requirement as above; however, AVs are less battery constrained so they can afford to spend more energy per bit.

Scenario 3: A radio on an AV needs to achieve 5 Mbps instead of 3 Mbps on the uplink in every 10-ms time window, to support, for example, Full HD (1080p) real-time video conferencing.

4.1 SDR as a reinforcement learning problem

The SDR learning engine uses RL to learn a control algorithm. We start by describing how the design specifications for the above scenarios may look like, and how they map to the RL framework.

Reward function Radio designers need to specify the communication objective, based on the use case and the device category. In the RL framework, this defines the *reward function*. For example, in the above scenarios, a radio designer might want the control algorithm to optimize the following objective in every 10-ms window,

$$(NBS - NBS_{target}) - \epsilon \times EnergySpent$$

where NBS and $EnergySpent$ are the number of bits sent successfully and the corresponding energy spent, NBS_{target} is 30000 or 50000 bits corresponding to 10 ms times 3 or 5 Mbps respectively, and ϵ is a measure of the energy sensitivity of the device. As an illustration, ϵ may be 10 times lesser for AVs than for phones, e.g., 22.5 and 2.25 bits per energy unit for phones and AVs respectively[4]. The first component rewards the control algorithm for how much it exceeds the throughput target while the second component penalizes it for the energy spent, and ϵ controls the relative tradeoff between these two conflicting components.

Action space Radio designers need to specify the control knobs available, and their possible settings. In the RL framework, this defines the *action space*. For example, in the above scenarios, a radio designer might want the control algorithm to adapt two knobs every 1 ms - the *modulation order* and the *transmit power* (via baseband scaling) - while all other knobs are held fixed at their default settings. The modulation order controls the sending rate and therefore influences the throughput. The transmit power controls the

[3] Each DNN that we train in Section 4 is on the order of 10 kB, so we expect the combined memory footprint to be 1 MB or less.

[4] penalty for spending 1 energy unit = reward for exceeding target by 22.5 or 2.25 bits

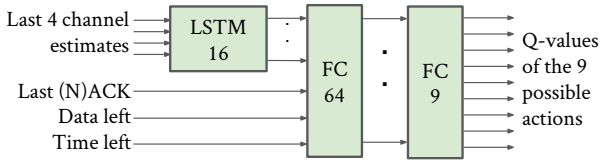

Figure 2: Architecture of the SDR Deep Q Network that we trained for each of the three reference scenarios in Section 4

energy spent, and also influences the reliability of the transmission, i.e., whether or not a transmission at a certain modulation order in a time step will experience a block error given the channel condition in that step. As an illustration, the possible settings may be 2 (QPSK), 4 (16-QAM) or 6 (64-QAM) for the modulation order[5], and 10, 15 or 20 dBm for the transmit power. The control algorithm's dilemma is that for a given transmit power, a higher modulation order in a time step can lead to a higher throughput *if* the transmission is successful, but also has a higher risk of a block error which would result in zero throughput in that step. On the other hand, for a given modulation order, a higher transmit power lowers the risk of a block error but also consumes more energy.

State space Radio designers need to specify the measurements available based on which the control knobs can be adapted. In the RL framework, this defines the *state space*. For example, the control algorithm could use the following measurements commonly available at a radio to adapt the modulation order and the transmit power: (i) *last* 4 *channel estimates*[6], (ii) *data left* to meet the throughput target in the 10-ms window, (iii) *time left* in the 10-ms window, and (iv) *ACK/NACK* of the transmission in the previous time step. While (i) and (iv) provide visibility into the link conditions, (ii) and (iii) help track the optimization state.

Learning environment Radio designers need to specify the wireless environment for which the control algorithm must be optimized. In the RL framework, this feeds into the *learning environment*. For example, a radio designer might choose the reference model in [13], and might want to optimize the control algorithm for mobility environments where speeds and path losses vary uniformly between 0-100 km/h and 70-90 dB respectively. Note that using such reference models is standard practice in the industry for designing and evaluating any new technology for mobile networks as it offers an easy, faithful and repeatable way for testing the technology in a wide variety of realistic and challenging deployment scenarios.

4.2 The SDR learning engine

In this section, we describe the working of the SDR learning engine, and explain how we converted the above specifications into a DNN-based physical-layer control algorithm for each scenario; see Figure 1 (central subfigure) for an illustration.

Physical-layer simulator We implemented a MATLAB simulator consisting of a physical layer that simulates an LTE- and NR-like OFDM-based transmitter and receiver, and a wireless layer that simulates the impact of wireless channels in baseband. In the wireless layer, we used 17600 of 22000 channel traces generated using the specified model above, each lasting 10 ms, for training the SDR control algorithm for each scenario; the rest were reserved for evaluation. In every step during training, the SDR control algorithm configured the modulation order and the transmit power of the physical layer, and the simulator fed back the updated state variables and the new reward.

RL method and DNN architecture We used deep Q-learning[7] to train the SDR control algorithm for each scenario. In this RL method, the control algorithm is represented using a Deep Q Network (DQN) [9] and it learns to estimate the *quality* of choosing each possible combination of actions from a given state. Figure 2 shows the architecture of the SDR DQN that we trained for each scenario. The last 4 channel estimates were fed into a Long Short-Term Memory (LSTM) layer, which is known to be good at time-series predictions, with 16 hidden units. The output of the LSTM layer was concatenated with the remaining inputs and fed through two fully-connected layers with 64 and 9 units each. The 9 outputs of the neural network correspond to the Q-values of the 3×3 possible combinations of actions.

Training We trained three SDR DNNs, one for each of the three reference scenarios. To make the training faster and more robust, we pre-cached the rewards and states from the simulator for all possible actions and time steps in the 17600 channel traces, and during training we only looked up a table for computing rewards and states. Exploration-exploitation tradeoff was annealed from 1 to 0.02 over 2 million time steps linearly, and learning rate was 0.01. Each DNN converged in around 20 million time steps, roughly corresponding to 2 million episodes, lasting about 32 hours[8] with a single Intel Xeon E5-2699 v3 CPU running at 2.30 GHz.

5 EVALUATION

In this section, we evaluate the SDR DNN-based physical-layer control algorithms that we trained in Section 4 for the three reference scenarios, in order to answer the following questions:

How does the SDR control algorithm perform compared to (i) LTE's uplink closed-loop power control and adaptive modulation algorithms, which represent the industry state of the art, and (ii) the offline-optimal control algorithm, which achieves the best possible performance in theory? (see Section 5.2)

What have the SDR DNNs actually learnt in terms of adapting to different scenarios and operating conditions? (see Section 5.3)

5.1 Evaluation method

Performance metrics The two metrics of interest to our reference problem are:

(i) the *target met (%)* which is the percentage of time windows in which the throughput target was met (higher is better), and

(ii) the *average transmit power* in a window (lower is better). Note that if the throughput target in a window is met in less than

[5]In the physical-layer setting that we use for our evaluation in Section 5, this translates to a sending rate of 2.1, 4.2 or 6.3 Mbps respectively.

[6]We arrived at the choice of 4 using trial and error, as adding more history did not provide any significant gain but made the training slower.

[7]There is scope to use more advanced learning algorithms, however as we show in Section 5, this learning algorithm already shows promising results.

[8]There is scope to make the training faster, however this is already a reasonable training cost given these models would be trained offline and would not usually need retraining for weeks or months.

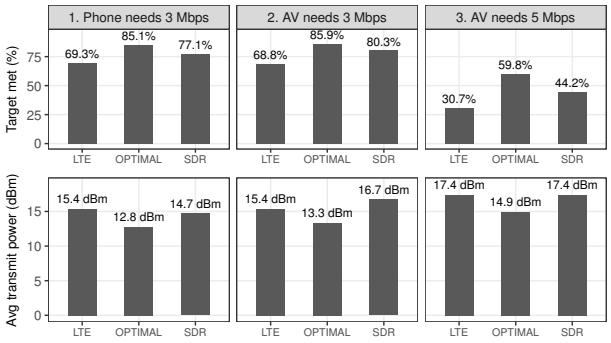

Figure 3: Benchmarking SDR's performance. In these scenarios, SDR meets its throughput targets 10-45% more often than LTE and 74-93% as often as the offline optimal solution, while using similar average transmit powers as LTE and 2-3 dB higher than offline optimal.

10 ms, the transmit power for the remaining time steps is assumed to be 0 while computing the average transmit power.

Benchmarks We evaluated three classes of control algorithms:

(1) *LTE*, specifically the adaptive modulation and power control algorithms defined in [2],

(2) *offline optimal*, obtained using dynamic programming assuming perfect knowledge of the future, and

(3) *SDR*, trained as described in Section 4.

Evaluation We evaluated the performance of the above control algorithms on 4400 traces, each lasting 10 ms, which were previously held back, and hence unseen, during the training of the SDR DNNs. Figure 3 summarizes the results.

5.2 Benchmarking SDR's performance

Note that the SDR approach is easily repeatable, as designing a control algorithm for a new scenario just requires us to provide the corresponding new specifications. In this section, we show that this repeatable approach is indeed able to adapt to new scenarios, and its performance is closer to optimal than traditional approaches.

SDR is able to adapt to different scenarios As one illustration, if we compare scenarios 1 and 2, we observe that SDR has been able to adapt to the lower energy sensitivity of AVs, as it uses a higher average transmit power for AVs (16.7 dBm) compared to phones (14.7 dBm) for the same throughput target (3 Mbps), to meet the target more often for AVs (80.3%) than for phones (77.1%). In contrast, since LTE does not adapt to the energy sensitivities of devices by design, its performance is similar (~69% target met using 15.4 dBm on average), and inferior to SDR[9], in both scenarios.

SDR's performance is closer to optimal than LTE In the three scenarios, SDR meets the throughput target 90.6%, 93.5% and 73.9% as often as the offline-optimal solution while using 1.9 dB, 3.4 dB and 2.5 dB higher transmit power respectively. In contrast, LTE meets the throughput target only 81.4%, 80.1% and 51.3% as often as offline optimal while using 2.6 dB, 2.1 dB and 2.5 dB higher transmit power respectively in the three scenarios.

[9]In scenario 2, even though SDR uses a slightly higher transmit power than LTE, it also meets the target more often; its overall reward is still higher.

5.3 Interpreting the behavior of SDR DNNs

In order to gain insights into what the SDR DNNs have actually learnt, we perform a *sensitivity analysis*. By this, we mean that we vary one input variable while holding the others fixed at different levels, and observe how each DNN changes its decisions in response. This allows us to both sanity check as well as interpret the behavior of the blackbox SDR DNN models.

In this section, due to lack of space, we restrict our attention to studying the sensitivity to the *last (most-recent) channel estimate* alone, which is the most important input for any adaptive modulation and transmit power selection algorithm. Figure 4 visualizes the results of this analysis in 6 different contexts. In each of the 6 subfigures, we sweep the most-recent channel estimate along the x-axis, from −95 dB to −70 dB, while the communication scenario and other input variables, indicated in the title and the subtitle respectively of each subfigure, are held fixed at different values.

Sanity check Qualitatively, we expect that as the link becomes stronger, the optimal behavior would be to transmit at higher modulation orders using lesser transmit powers. In each of the 6 subfigures, we observe that the SDR DNNs have been able to learn the expected qualitative behavior, i.e., as the most-recent channel estimate increases, the modulation order increases (monotonically) and the transmit power reduces (albeit non monotonically[10]).

Insights So what did the DNNs additionally learn that enabled them to outperform LTE? We observe that they have learnt to adapt the answers to three broad questions based on the context.

(1) What are the channel estimate thresholds for switching up the modulation order from 2 to 4, and from 4 to 6?
(2) Should the transmit power also be switched up when the modulation order switches up, and by how much?
(3) What are the channel estimate thresholds for switching down the transmit power?

We encourage the readers to study the subfigures in Figure 4 and observe how the answers to the above questions change depending on the context, specifically how they are different for different (i) devices (compare subfigures 1 and 2), (ii) throughput targets (compare 2 and 3), (iii) data left (compare 1 and 4), (iv) time left (compare 3 and 6), and (v) channel gradients (compare 2 and 5). These differences reveal why it is extremely hard for an algorithm based on manually-specified rules and thresholds, like the control algorithms used in LTE today, to capture the optimal solution, and why it is instead better to use a deep learning-based approach to design control algorithms for next-generation radios.

6 TAKING SDR TO PRACTICE

Where will the SDR learning engine operate in practice? We expect that the SDR learning engine will operate as a service, either locally at each base station or in the cloud, the latter being easier to scale and manage. Whenever a new scenario needs to be supported, a radio designer would submit the new design specifications to this service, which in turn will train an appropriate DNN and push it to the relevant base stations. Whenever the specifications change

[10]Whenever SDR increased the transmit power even as the link became stronger, it was always to support a corresponding increase in the modulation order so as to ensure the transmission is still reliable, which is a sensible behavior in hindsight.

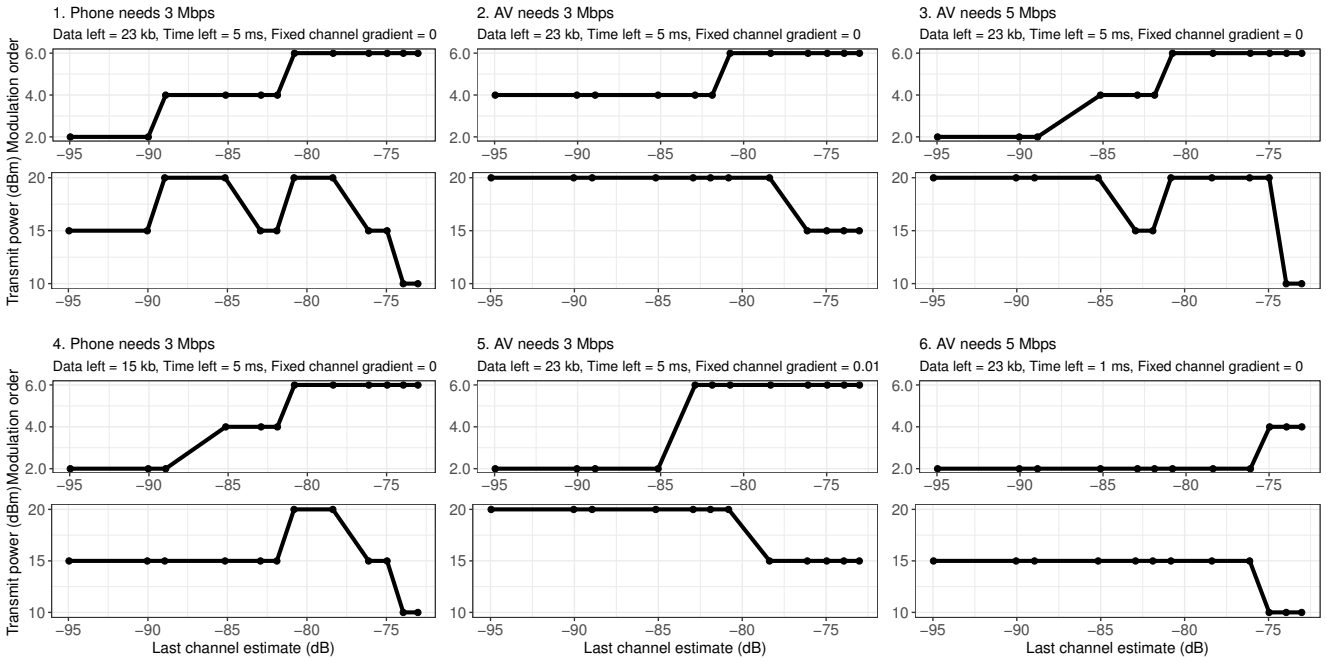

Figure 4: Sensitivity of the SDR deep neural networks to the most-recent channel estimate in different contexts. In all subfigures, the most-recent channel estimate is swept along the x-axis from −95 dB to −70 dB while the communication scenario and other input variables (indicated in the title and the subtitle of each subfigure respectively) are held fixed at different values.

or a base station detects a drift in the wireless environment, this service will be requested to retrain the affected DNNs.

What support is needed from the base stations? In order to use the SDR approach, base stations need to have (i) the flexibility to download and use new or upgraded control algorithms for different scenarios, and (ii) the ability to run the SDR DNNs in real time. The DNNs may need to run as often as every 1 ms in LTE, and possibly more often in NR; the DNNs that we trained in Section 4 took 0.8 ms on average to execute on a standard MacBook Pro with 2.9 GHz Intel Core i5. We envision that the base stations will use hardware accelerators to run these DNNs, which can reduce this by an order of magnitude and comfortably meet the real-time requirement.

How does the SDR approach work when a base station has to serve multiple devices simultaneously? This is a challenging open question and requires the SDR approach to be extended to the link layer, i.e., resource scheduler. This requires expanding both the state space (to handle states of a variable number of devices) and the action space (to also select resource schedule in addition to the physical-layer control knobs), incorporating fairness criteria in the objective function, and training the SDR DNNs using a simulator that also simulates the link-layer dynamics.

Acknowledgment We would like to thank our reviewers and our shepherd for their valuable reviews and feedback that helped bring this draft to its current state. We would also like to thank Manikanta Kotaru, Aaron Schulman and Kanthi Nagaraj for reviewing the draft, and Stanford Platform Lab for supporting this project.

REFERENCES

[1] 3GPP. NR: Physical channels and modulation. TS 38.211, Mar. 2018.
[2] 3GPP. NR: Physical layer procedures for control (Release 15). TS 38.213, Mar. 2018.
[3] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti. Cellular network traffic scheduling with deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.
[4] R. Daniels and R. Heath. Online adaptive modulation and coding with support vector machines. In *IEEE European Wireless Conference 2010*, pages 718–724.
[5] Y. He, C. Liang, F. R. Yu, N. Zhao, and H. Yin. Optimization of cache-enabled opportunistic interference alignment wireless networks: A big data deep reinforcement learning approach. In *IEEE International Conference on Communications (ICC) 2017*, pages 1–6.
[6] ITU. IMT Vision: Framework and overall objectives of future development of IMT for 2020 and beyond. Technical report, Sept. 2015.
[7] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with Pensieve. In *Proceedings of SIGCOMM '17*, pages 197–210, New York, NY, USA.
[8] F. Mismar and B. Evans. Q-Learning Algorithm for VoLTE Closed-Loop Power Control in Indoor Small Cells. *arXiv*, July 2017.
[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.
[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.
[11] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *IEEE Conference on Computer Communications, INFOCOM 2017*.
[12] Z. Wang. The applications of deep learning on traffic identification. *BlackHat USA*, 2015.
[13] C. Xiao, Y. Zheng, and N. Beaulieu. Novel sum-of-sinusoids simulation models for Rayleigh and Rician fading channels. *IEEE Transactions on Wireless Communications*, 5(12):3667–3679, December 2006.
[14] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy. A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs. In *IEEE International Conference on Communications (ICC) 2017*.
[15] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. *arXiv preprint arXiv:1803.04311*, 2018.