Learning Sensor Control for Information Gain in Dynamic, Partially Observed and Sparsely Sampled Environments

J. Brian Burns Aravind Sundaresan Pedro Sequeira Vidyasagar Sadhu
SRI International, Menlo Park, CA 94025, USA
{brian.burns,aravind.sundaresan,pedro.sequeira,srikanthvidyasagar.sadhu}@sri.com

ABSTRACT

We present an approach for autonomous sensor control for information gathering under partially observable, dynamic and sparsely sampled environments that maximizes information about entities present in that space. We describe our approach for the task of Radio-Frequency (RF) spectrum monitoring, where the goal is to search for and track unknown, dynamic signals in the environment. To this end, we extend the Deep Anticipatory Network (DAN) Reinforcement Learning (RL) framework by (1) improving exploration in sparse, non-stationary environments using a novel information gain reward, and (2) scaling up the control space and enabling the monitoring of complex, dynamic activity patterns using hybrid convolutional-recurrent neural layers. We also extend this problem to situations in which sampling from the intended RF spectrum/field is limited and propose a model-based version of the original RL algorithm that fine-tunes the controller via a model that is iteratively improved from the limited field sampling. Results in simulated RF environments of differing complexity show that our system outperforms the standard DAN architecture and is more flexible and robust than baseline expert-designed agents. We also show that it is adaptable to non-stationary emission environments.

KEYWORDS

Reinforcement Learning, Partial Observability, Dynamic Sparse Environment, Sensor Control, Information Gain, Model-based

1 INTRODUCTION

Overview. Sensor control that maximizes information gain under partially observable and dynamic environments is an important problem that has several applications [18]. For example, consider the problems of tracking widespread activity from a limited, but controllable field of view, or tracking carbon monoxide levels over a geographic area with the minimum number of sample sites. A relevant problem, which we study in this work, is Radio Frequency (RF) spectrum monitoring, which involves detecting and tracking multiple dynamic signals in a potentially large RF spectrum using an RF receiver (sensor) with only a limited, but tunable, reception band. All these problems require a sequential decision making approach that indicates which sensor(s) or sensor settings to choose at each time instant based on history to maximize information gain e.g., knowledge about the number of signals, emitting bands, intervals of activity, communication patterns, etc.

Challenges. The above tasks present several challenges: (i) *partial observability*: each sensor (or sensor band) can only provide partial observations of the underlying state; (ii) *dynamic environments*: the underlying environment is stochastic and could be non-stationary, making it hard to track its state over time; (iii) *sparse environments*: it is possible that useful information may only be infrequently collected, meaning that the majority of observations are mostly non-informative; (iv) *costly samples*: taking samples from an actual fielded sensor is costly and so interactions with it are restricted according to some *budget*.

Approach. In this paper, we study and evaluate our approach for an abstracted RF spectrum monitoring task. In particular, the goal is to control a band-limited RF receiver to optimize multi-signal detection and tracking throughout an extensive RF environment. The autonomous controller is given a range of the spectrum (discretized into several frequency bands) within which to operate. It is not given information about the specific frequencies, densities or distributions of the signals within this range. At each instant, it can sample a sub-band of the spectrum to observe if there are signals present in that sub-band. We developed an RF simulator on which a controller is trained to decide *which* sub-band to sample at each instant to accomplish the goal of tracking/predicting the signals in the full spectrum. During the training process, the controller learns the general behavior of the signals, such as frequency switching patterns and typical durations, and learns how to use this knowledge to search for and track these types of signals in novel situations.

Towards this goal, we propose an approach based on the Deep Anticipatory Network (DAN) [18] framework for optimal sensor control in information gathering tasks. DAN uses deep Reinforcement Learning (RL) with prediction rewards, where the action suggested by a *value-network*, that implements the value function, is rewarded whenever a separate *model-network* makes a correct prediction of the true state based on observations that are the result of the action. In order to scale to large problem spaces, e.g., large spectrum and/or small reception bandwidth, and to efficiently learn and process signals with potentially complex frequency-time patterns, we extended the DAN approach by implementing the prediction and control functions using neural networks with hybrid convolutional-recurrent layers, such as the ConvLSTM layer [24]. Additionally, as the environment is dynamic and sparse, we propose novel information-gain rewards for our RL approach to encourage exploration thereby avoiding sampling regions of spectrum where signals were already identified.

Another important aspect of our approach, that was not considered in the original DAN work, is the use of potentially limited data from a fielded controller (referred to as *experience feedback*) to update the RF simulator used for training. In particular, a controller is trained in simulated RF environments by using a simulator that parameterizes a distribution over several aspects controlling the dynamics of the RF spectrum, e.g., the number of emitting signals, the frequency at which they emit, different communication patterns, etc. We refer to this as the *lab* simulator. Using it, the controller can train on many and diverse samples of environments — far more than can be sampled out in the actual field environments over a reasonable training period. However, in a real-world situation, the distribution over the actual field RF environments of interest may be non-stationary, which can make the conditions in which the controller was trained obsolete and consequently the controller sub-optimal. In this work, we investigate how the lab simulator can be updated using field experience data gathered by deploying the trained controller in the field for a limited amount — the interactions are restricted according to some budget. These field samples are used to adjust the lab simulator's parameters such that the generated environments closely match those encountered in the field, and the controller's policy can be fine-tuned by RL-based retraining on the updated lab simulator. We study this problem by using an additional *field* simulator — one that holds the ground-truth of the distribution over the environment parameters — to simulate the collection of limited field data and to test the retrained system.

In this work, we will refer to a stochastic model of an RF environment, which determines its dynamics, as a *spec*. The field simulator will have a *field spec*, which is hidden to the system, and during any given training iteration, the lab simulator will have a *lab spec* that is used to train the controller. In addition to estimating the field parameters/spec using field samples, we study an alternative approach of learning the field state dynamics model directly using field samples. We then use this field state model in the lab as an environment to train a controller using model-based RL.

Contributions. Our contributions in this paper are as follows:

- Extending the DAN framework to handle the challenging problem of RF monitoring by scaling up the control space and enabling the monitoring of complex, dynamic activity patterns using hybrid convolutional-recurrent processing steps.
- Improving exploration in sparse, non-stationary environments using novel information gain rewards.
- A supervised learning variant that is shown to perform better than RL approaches in certain environments.
- Two model-based RL approaches capable of model updating using experience feedback from limited field deployment.
- An evaluation of the the proposed approaches using various architectural configurations showing their benefit over baseline controllers in different environments.

2 RELATED WORK

General approaches. Information gathering under partial observability, which is related to Partially Observable Markov Decision Processes (POMDPs) [8], has been explored in several domains. For example, Satsangi et al. [18], present a technique (our baseline approach) for information gathering using Deep Anticipatory Network (DAN) that decides which camera to switch on to track people in a mall. They also applied a related approach to active perception in robotics [19]. Wang et al. [22] propose an RL technique for information gathering using sparse mobile crowd-sensing that tells which cell to collect data from to minimize the uncertainty in state estimation. Mnih et al. [14] and Haque et al. [4] present an image classifier that adaptively selects regions for processing at high resolution. Most of these techniques do not address our

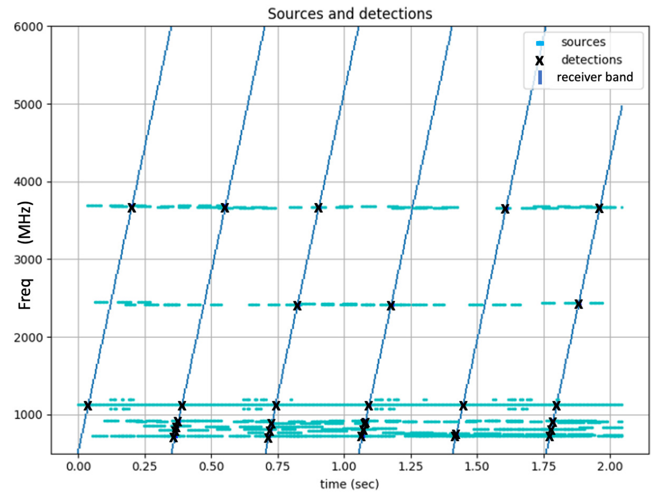


Figure 1: A partially observed RF environment with sequential scan for control, which misses the short signal bursts at 1.2GHz. Plot: time(secs) vs frequency(MHz); cyan: signals; dark blue: sampled bands; black x marks: detected signals.

problem combining dynamic environment, reward sparsity and limited field experience. In the domain of generic POMDP solutions, James and Singh [7] present an algorithm for learning in POMDPs assuming availability of 'landmarks' or special states that are not hidden to the agent, but not available in our domain. Katt et al. [10] present an approach based on Monte-Carlo tree search and look-ahead planning for solving POMDP problems in an online manner. However, due to large dimensional state space, such an approach is not scalable to our domain.

RF domain. In the domain of RF spectrum monitoring, most of the existing work is on signal detection and identification (SDI) rather than tracking. Franco et al. [2] present a hierarchical two-step approach for SDI. At a finer scale, they first use a sweeping window to detect the local presence of signals, and at a larger scale, these local detections are integrated using a frequency-time region proposal network. Kulin et al. [11] present a single-step, an end-to-end deep learning approach for wireless signal detection. Mendis et al. [13] present an attention-driven RL technique for signal detection in a wide-band spectrum. The proposed method consists of two main components: a spectral correlation function (SCF) based spectral visualization scheme and a spectral attention-driven RL mechanism that adaptively selects the spectrum range and implements the intelligent signal detection. This approach however assumes that the modulation technique is known *a priori*.

Experience Feedback. This problem is related to other types of machine learning tasks. In transfer learning for RL, (Taylor and Stone [20] and Zhu et al. [25]) there is a source (lab spec) and target (field spec) population; however, partial observability is not addressed. In model-based value expansion (imagination rollouts), the number of real (field) interactions is reduced by doing additional training with updated simulator (Feinberg et al. [1]). Kalweit and Boedecker [9] and Hafner et al. [3] discuss RL imagination rollout approaches based on uncertainty and latent space. However, unlike our problem, most of these works deal with continuous actions. Model-based RL traditionally involves interactions between a planning module and RL training (Moerland et al. [15]); Li et al. [12] discuss a technique for accelerating model-free RL using imperfect models for the related application of spectrum access. Hua et al. [5] present a GAN-based method to learn the state-action values for resource management in network slicing. Huang et al. [6] present a Generative Adversarial Interactive Reinforcement Learning that combines the advantages of GAN-based learning and interactive RL. In our problem, the field state to be learned is a time-series system and most existing approaches do not handle this case.

3 PROPOSED SOLUTION

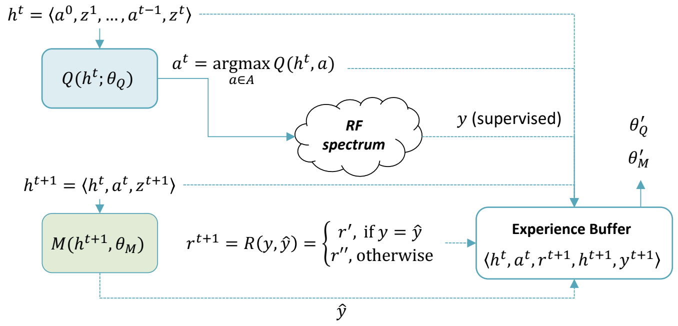
3.1 State Estimation and Tracking

The problem involves finding and tracking multiple signals that have activity patterns of varying complexity involving several unknown parameters. A controller agent is trained and tested using multiple episodes (trials), where each episode has a new signal environment sampled from the same stochastic environment model (*spec*), and the episode involves a sequence of control (frequency selection) and observation (signal detection) steps. The specific signal patterns of each new environment are unknown to the agent and must be inferred through tracking and monitoring during the interaction. See Figure 5 for an overview of two sample episodes. To solve this problem, we extend the original DAN architecture and rewards.

Problem Formulation. Following the DAN approach in [18], we consider a Partially Observable Markov Decision Process (POMDP) [8] to model the dynamics of our RF environment. We denote $s \in S$ to represent the hidden state of the environment (all the signals in the environment along with their center frequencies), $y \in Y$ to denote a target variable of interest (whether a signal is present in each band) that depends only on s, $z \in \Omega$ to denote a partial observation (whether a signal is present in the observed sub-band) that is correlated with y and $a \in A$ to denote the action (which sub-band to observe) taken by the agent. At each discrete timestep, t, the agent takes an action a^t, the environment transitions to state s^{t+1} and the agent receives an observation z^{t+1}. The goal of the agent is to correctly predict the target variable y^t given the history of previous actions and observations denoted by $h^t = (a^0, z^1, ..., a^{t-1}, z^t)$. We denote by $\hat{y}$ the agent's prediction of y. At each step, the agent receives a reward, denoted by $r(y^t, \hat{y}^t)$, that indicates how similar y^t and $\hat{y}^t$ are.[1]

RF Environment. In the context of RF spectrum monitoring, s represents the state of the whole RF spectrum, that contains a set of unknown *entities* interacting with each other and transmitting signals spread across a range of frequencies band and varying in time. Further, y is the vector representing signal activity (0 for none and 1 for activity) at all frequency bands, a represents the frequency band(s) that the agent samples, and z represents observed detections (the presence/absence of signals at the sampled bands); h is the history of sampled frequencies and observations. The agent's actions can only sample the spectrum at specific frequencies. The goal of the agent is to select actions at each timestep in order to

[1]The state (s) and target variable (y) are, in effect, equivalent in our experiments and hereafter synonymous.



$h^t = \langle a^0, z^1, ..., a^{t-1}, z^t \rangle$
$Q(h^t; \theta_Q)$
$a^t = \operatorname{argmax}_{a \in A} Q(h^t, a)$
RF spectrum
y (supervised)
θ'_Q
θ'_M
$h^{t+1} = \langle h^t, a^t, z^{t+1} \rangle$
$M(h^{t+1}, \theta_M)$
$r^{t+1} = R(y, \hat{y}) = \begin{cases} r', \text{if } y = \hat{y} \\ r'', \text{otherwise} \end{cases}$
Experience Buffer
$\langle h^t, a^t, r^{t+1}, h^{t+1}, y^{t+1} \rangle$
$\hat{y}$

Figure 2: Deep Anticipatory Network (DAN) overview.

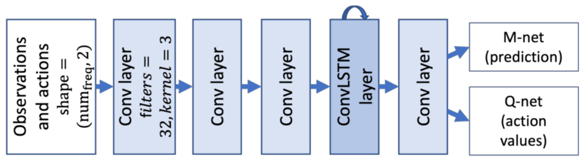


Figure 3: ConvLSTM-DAN architecture with Q and M outputs. The ConvLSTM is convolutional in frequency and recurrent in time.

accurately report the presence/absence of signals in *all* frequency bands at that timestep. Fig. 1 shows an example of a partially observed RF environment. One or more bands can be active (contain signals) at any given time. Here, a sequential scan is performed across the spectrum, which misses the short signal bursts around 1,200MHz.

Environment Spec Details. An environment spec defines an environment population model as a range of possible values for the following parameters: (i) *number*: number of interacting signal pairs; (ii) *width*: total number of spectral bands spanned by the pairs; (iii) *period*: period of the signal pair interaction; (iv) *duty cycle*: duration of one signal over the other during an interaction cycle; (v) *frequency*: the frequency of the lowest of the signal pair; (vi) *start*: timestep when signal pair appears in spectrum. Table 1 lists some of the environment specs and parameters used in our experiments.

3.2 DAN framework

Deep Anticipatory Networks (DANs) [18] were developed for information-gathering tasks under partial observability e.g., tracking people using cameras. A DAN makes control decisions that maximize the information gathered (minimize the uncertainty) about a target variable of interest y that cannot be fully-observed. The key point is that, for this type of task, DAN avoids complex belief state updating and instead uses state prediction rewards to guide the agent behavior.[2] DAN uses two neural networks to train a policy, as shown in Fig. 2. The first, Q, takes the history of previous

[2]This has been demonstrated in Satsangi et al. [18] to be equivalent in effect to belief updating.

actions and observations, h_t, and produces Q-values for the different actions. A second network, $\boldsymbol{M}$, takes the history h_t plus the last action and observation to predict $\hat{y}$, i.e., an estimate of target variable y. M is trained in a supervised manner with ground truth data, i.e., using the actual target values y. Q is a standard value function network trained using RL but here it is rewarded when M's predictions are accurate, i.e., when $\hat{y}$ is similar to y. Q is trained to maximize cumulative discounted reward, $\sum_t r_t \gamma^t$. The Q and M networks are trained simultaneously and, after training, only Q is used for autonomous control for information gain. In all of our DAN variants, M is trained using weighted binary cross entropy (WBCE) as a loss, weighted to emphasize our metrics; see the discussion on rewards and metrics in Sec. 3.2.2.

3.2.1 ConvLSTM, Predictive and InfoMax DAN Architectures. The problem of RF spectrum monitoring presents several challenges, including: being able to scale to large discrete frequency and time domains, identifying repeating signal patterns in frequency-time space and coping with non-stationary nature of the environment. The original DAN does not address these challenges; we propose various architectural and reward modifications to address them.
ConvLSTM-DAN. The baseline DAN uses fully connected/dense layers for both Q and M networks with Rectified Linear Unit (ReLU) activation, which doesn't scale effectively and makes learning of frequency-invariant activity patterns difficult. We address the scalability of the problem space by using convolutional layers (convolution in frequency). We also introduce hybrid convolutional-LSTM (Long Short Term Memory) / ConvLSTM layers [24], which combine convolution in frequency dimension and recurrence/memory in time dimension for learning translation-invariant frequency-time activity patterns. This architecture is shown in Fig. 3. For ConvLSTM training, we use a reward that is a function of the intersection over union (IoU) of the predicted output from M and the ground truth; see the discussion on rewards and metrics in Sec. 3.2.2. As the input to both Q and M networks are similar, and action selection and state prediction are related tasks, the layers used to compute features can be shared to reduce the training parameters. We call this enhanced framework *ConvLSTM-DAN*, which has two outputs, one for Q and the other for M. (All of our DAN variants discussed in this paper use a shared architecture.) After studying the effect of different well-known DQN/RL enhancements, we found that Dueling DQN [23] improved results across the board and we include it in our standard ConvLSTM-DAN framework (not shown in Fig. 3): for Q, the output of the last convolutional layer is split into two streams — Value and Advantage — which are then combined to give the Q output. Double DQN [21] was not considered as useful and was not incorporated.
Predictive DAN. If the environment spec defines a variable number of signals it introduces an interesting exploration / exploitation challenge with subtle differences in the payoff odds between carefully tracking the signal found vs. finding a new one.This problem is exacerbated in reward-sparse scenarios (with an empty spectrum most of the time). In order to encourage exploring an unknown state space and at the same time exploit the known state space, we propose an enhancement over the ConvLSTM-DAN that provides an auxiliary predictive reward, which we refer to as the Predictive-DAN. This is similar to the Intrinsic Curiosity Module [16]. In Predictive-DAN, the shared network in Fig. 3 has 3 outputs: m-output, $M = y_t|h_t$ (the current state given current observation history), q-output, $Q = a_{t+1}|h_t$ (next action to take given current observation history) and a predictive output, $\boldsymbol{P} = y_{t+1}|h_t$ (the next state given current observation history). Here, state refers to target variable, y. When the next action, a_{t+1} is actually taken, then the m-network estimates the current state given current information: $y_{t+1}|h_{t+1}$, where $h_{t+1} = \{h_t, a_{t+1}, z_{t+1}\}$. We can denote the information gained from this action as $infogain = mean(abs(y_{t+1}|h_t - y_{t+1}|h_{t+1}\|))$, which is used as an auxiliary reward to train the Predictive-DAN Q-network: IoUreward + infogain $*$ 10. In practice, typically only the observed 'band' will change if at all, but if the network has learned relations between signals in other bands, that could be affected too. This is similar to the reward that can be used when there is no ground-truth (the difference being that this is only computed for the observed state and not full state).
InfoMax-DAN. While in above architectures, we compute and use information gain only for the executed action, it is also possible to compute the information gained for all actions instead of just the action taken. We use this idea to train Q-net in a fully supervised manner without using RL. This approach maximizes information gained in a single step (with no cumulative reward). We refer to this alternative approach as InfoMax-DAN, which is fully supervised version of Predictive-DAN.

3.2.2 Rewards and Metrics. Since the goal in our control problem is to maximize information about signal presence, it seems reasonable to start with the commonly used detection/localization metric Intersection over Union (IoU). In our case, *intersection* is the count of (time, frequency) positions where true signals coincide with predicted ones, and *union* is the total count of true positions plus the total for predicted ones (where predictions can either be probabilities or thresholded, binary states). We use the following variants of IoU for rewards and metrics in our experiments: *Instantaneous IoU* is IoU for one slice of time; *Cumulative IoU* is cumulative IoU up to a given time; *Block IoU* is cumulative for the last N timesteps. The Instantaneous IoU can provide frequent rewards but is unstable in sparse environments, which be can addressed using a differential reward: *Differential Block IoU* = $BIoU_t^{N+1} - BIoU_t^N$, where the $BIoU$ are block IoU, for blocks of fixed N. For *losses* to train prediction, we use weighted binary cross entropy (WBCE), which is an effort to tilt the commonly used BCE more towards detection of sparse signals: the error for existing signals is weighted more than the error for non-existing signals. (The weight for bands *without* signals is set to the actual signal density and the weight for bands with signals is set to 1 − signal density.) In experiments, we have noticed that WBCE is qualitatively better than BCE or IoU for a single time-step which is ill-defined for a single time-step.

3.3 Experience-feedback via Model-based RL

Often, the *lab spec* used for training does not entirely cover the field (ground-truth) population of environments — for example, the domain knowledge employed may be outdated. To address this challenge, we introduce an experience feedback loop. Once trained on the lab spec, the agent is then deployed in field environments (here, simulated by a *field spec*) for a limited time, during which it collects samples that might be sparse. In this fielded phase, unlike

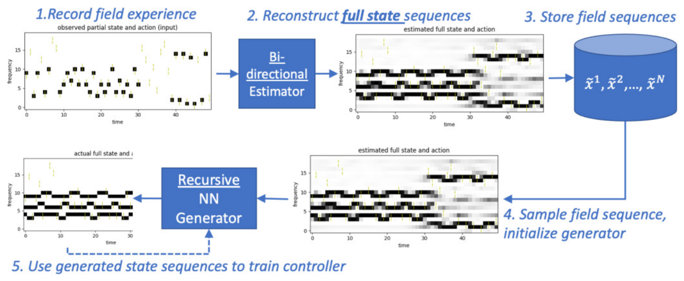

1.Record field experience
observed partial state and action (input)
2. Reconstruct full state sequences
estimated full state and action
3. Store field sequences
Bi-directional Estimator
$\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^N$
estimated full state and action
4. Sample field sequence, initialize generator
Recursive NN Generator
5. Use generated state sequences to train controller

Figure 4: Experience feedback loop via direct field state estimation.

the lab training, the agent does not have access to the underlying full state required to train the M network via prediction loss. Rather, these experiences are used to estimate the environments' dynamics, which are then used to update a lab spec/model. The agent is then retrained using the updated spec/model and re-deployed for more samples, and the process repeats for a number of times. We develop two model-based RL approaches to implement this feedback loop: (i) *field spec estimation* uses collected experiences to estimate the field spec parameters directly, which is then sampled to retrain the controller; (ii) *field state estimation* uses them to build a deep, generative model of the field state sequence population, from which we sample to retrain the controller.

3.3.1 Field Spec Estimation. Experience feedback using field spec estimation uses the following basic loop: (i) deploy a controller in the field (simulator with field spec), (ii) use an expert system to estimate the field parameters from the resulting samples (Sec. 4.1), (iii) add the resulting *estimated* field spec to a pool of training specs (including original lab spec), (iv) train the DAN controller using samples from the training pool of specs, and (v) deploy the resulting trained DAN. We studied three different variations of these steps. **Estimating Spec from Feedback:** in step (i), deploy our expert controller (Sec. 4.1), and in step (iv), train from only the estimated field spec. **Estimating Spec for Retraining:** in step (i), deploy a DAN controller trained on a general lab spec, and in step (iv), train from the spec pool with sample rates: estimated field spec=0.7, lab spec=0.3 (to avoid forgetting). **Bootstrapping:** do multiple iterations of the latter scheme, varying the field spec each deployment and training from all previously estimated specs.

3.3.2 Field State Estimation. One drawback of the Field Spec Estimation approach is that it assumes that observed dynamics in the field can be modeled by fitting a predetermined set of (known) simulator parameters. In realistic settings, we cannot anticipate all parameters governing the observed behavior in the RF spectrum — e.g., what if the entities are using a different communication protocol? To address this challenge, we follow a different approach to experience feedback that retrains the agent using Machine Learning (ML)-based methods that do not rely on parameterized environment specifications. Instead, the approach discussed here trains the controller on full, extended state sequences reconstructed and extrapolated from partially observed field samples. This will be evaluated against one that directly does controller fine-tuning on the raw, partially observed samples.

In *field state estimation*, we first train our DAN controller using a generic Lab Spec. We then adopt the following procedure (Fig. 4): (i) deploy and collect field experiences, which are sequences of partially observed states; (ii) reconstruct full state sequences from the partially observed samples using a bidirectional Recurrent Neural Network (RNN) or U-net architecture [17]; (iii) store reconstructed field sequences in a database; (iv) sample these stored sequences and use to initialize an RNN-based generator that then emits extended sequences; (v) binarize the generated sequences (1=signal, 0=none); (vi) use these to train the controller, instead of a parameterized spec.

This will be compared to a DAN controller that is fine-tuned in the field directly (no lab retraining) using the available *partially* observed field states (see 1. in Fig. 4). As we have ground truth only in the particular band sampled, the reward obtained by the M-network was modified to be only the prediction in this band.

4 PERFORMANCE EVALUATION

Experimental Setup. In order to have sufficient complexity in environment and control space to test our proposed approach, we considered environments that vary in terms of the number of communicating signal pairs and their properties; Table 1 shows some of the environment specs studied. We will start in Section 4.1 by comparing our DAN controller with expert-designed systems and study their adaptability using SpecA as the environment for which the expert-designed systems are optimized, while SpecB1 and SpecB2 – environments with increased variation and complexity – are used to test the adaptability. Example episodes from SpecA and SpecB2 are shown in Fig. 5. Then, in Sec. 4.2, we will extend the study to include non-stationary, semi-periodic and wider spectrum environments with multiple signal classes, such as in Fig. 1.

4.1 Comparison with non-ML Controllers

We hand-coded four controllers that follow simple, but often effective, behavior rules for RF environments to serve as baselines to evaluate the DAN-based system robustness:

Random: randomly selects bands in the spectrum. For prediction, it uses a *persistent state* scheme: each band's signal state (active/inactive) is set to the last observation in it. Initially, all bands are considered inactive.
Scan: sequentially selects bands in the spectrum, uses the same persistent state scheme for prediction.
Scan-and-Dwell: estimates the different spec parameters governing the dynamics of the environment from the sampled observations. The controller is given ranges for the parameters but does not know their true values. It traverses the spectrum sequentially and whenever an action triggers a signal detection, it subsequently samples the same band until the true parameters controlling the dynamics in *that* band are learned, via a process of elimination. The predicted state is computed for each from the estimated parameters.
Expert: uses the same process as the above controller to estimate the spec parameters from experience. At each step, it selects the band with the highest associated uncertainty, i.e., whose current range of possible values is the largest.

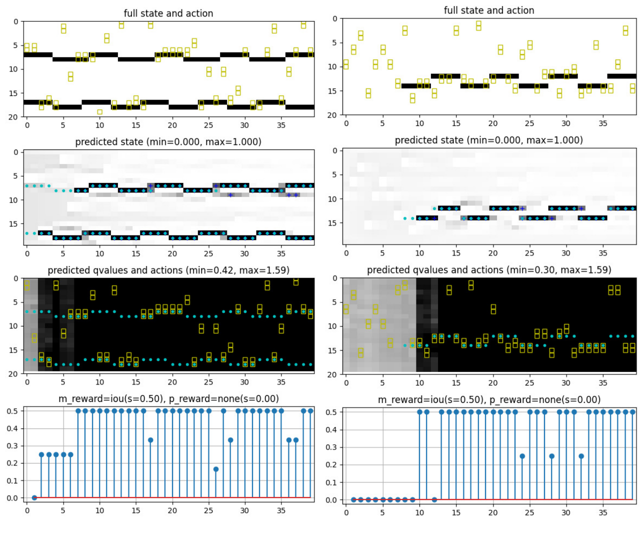
In particular, the Scan-and-Dwell and Expert controllers serve as reasonable upper bounds on performance since in our experiments

Params	A	B1	B2	C1	C2	F1	F2	F3
Number	2	[1, 2]	[1, 2]	[1, 2]	2	1	2	[1, 2]
Width	2	[2, 3]	[2, 3]	3	2	3	2	[2, 3]
Period	[8, 9]	[8, 9]	[8, 9]	[8, 9]	[6, 9]	[8, 9]	[8, 9]	[6, 7]
DutyCyc	4	[4, 5]	[4, 5]	4	[2, 5]	4	7	[3, 4]
Band	Rand	Rand	Rand	Rand	Rand	Rand	Rand	Rand
Start	0	0	[0, 10]	0	0	0	[5, 10]	[0, 5]

Table 1: Environment specs: The parameters are defined in Sec. 3.1; E.g. For Spec A, the number of signal-pairs is 2, the width between signal pairs is 2, the period is randomly chosen from [8,9], the duty cycle is 4, the band of each signal-pair is randomly chosen and the signal always starts at time 0. [a,b] means random inclusively within range [a,b].

	Environments		
Agent	Stat.	Non-stat.	Multi
ConvLSTM-DAN w/ db_iou	0.62	0.37	0.50
ConvLSTM-DAN w/ in_iou	0.63	0.38	0.48
Predictive-DAN	0.68	**0.45**	0.44
Infomax-DAN	**0.75**	0.39	**0.52**

Table 2: Performance of different DAN configurations and rewards, per environment (best in bold): scores are cumulative IoU, db_iou is differential block IoU reward, in_iou is instantaneous IoU reward (Sec. 3.2.2). Stat. refers to a stationary environment, where signals can appear at any time.



(a) SpecA (b) SpecB2

Figure 5: Training episodes from SpecA (a) and SpecB2 (b). First row: the full state (bands with signal activity are black and actions taken are gold squares). Second row: predicted state (*M*-net output) with signal probability in gray-scale (white = 0, black > 0.5) and blue dots are real signal. Third row: *Q*-net output (q-values) in gray-scale with action taken (gold boxes) and true state (blue dots). Fourth row: IoU rewards per step, where IoU measures predicted state – true state match.)

they are given relatively small ranges for the different spec parameters around their ground-truth values. In other words, an ML agent that has been trained in environments whose spec parameters are similar to those governing the environment on which it is being evaluated can be expected, at best, to attain a performance similar to that of the best hand-coded controller.

Fig. 6 compares the performance of hand-coded controllers and ConvLSTM-DAN trained in only one environment spec (SpecA) and tested in multiple environment specs (SpecA and SpecB1). In Fig. 6a the controllers were tested in environments from SpecA. The Expert and Scan-and-Dwell controllers (which are given SpecA

Model	Sep. Dense	Shared Dense	Shared Conv.
IoU	0.17	0.25	**0.35**
Parameters	82216	89858	**39234**

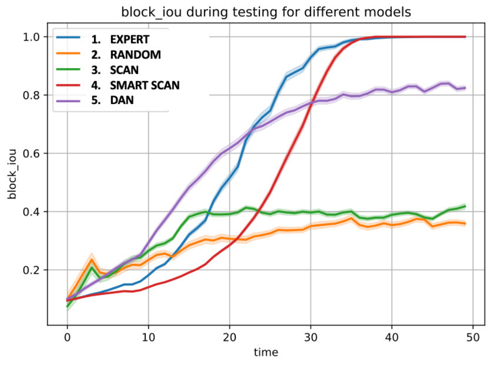
Table 3: Our Shared ConvLSTM model converges faster to a higher accuracy (IoU) than original DAN models. Both shared models have more layers than Sep.Dense

to estimate each sampled environment parameters) correctly predict the signal from $t = 35$. ConvLSTM-DAN achieves a good performance because it was trained on SpecA environments. Then, when tested on SpecB1 environments while being given SpecA to estimate the parameters (Fig. 6b), the hand-coded solutions fail to correctly predict the out-of-distribution signals. In contrast, the ConvLSTM-DAN, trained only in SpecA, *shows more robustness to changes in environment dynamics.*

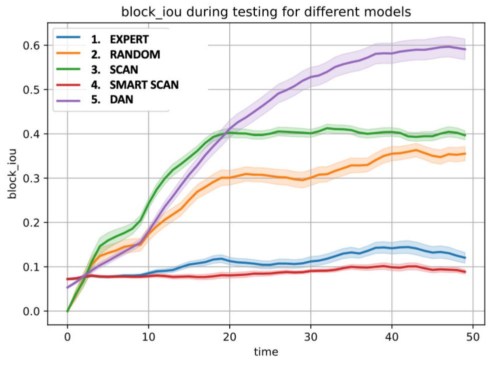
4.2 ConvLSTM, Predictive and InfoMax DAN

ConvLSTM-DAN vs. original DAN. We compare our ConvLSTM-DAN model (a shared architecture) with the models used in the original DAN: *Separate Dense* (*M*-net and *Q*-net are separate networks with dense layers followed by an LSTM layer) and *Shared Dense* (a single network with shared initial layers and separate outputs for *M*-net and *Q*-net). The ConvLSTM-DAN layers can scale arbitrarily with respect to number of frequencies without a corresponding increase in the number of model parameters and as a result it converges much faster to a better accuracy as shown in Table 3.

Predictive, InfoMax vs. ConvLSTM-DAN. We evaluate their performance in environments with distinct challenges: (i) Stationary (simplest): a range of patterns (period, duty-cycle) and signal numbers (1-3); (ii) Non-stationary (more challenging and realistic): narrower range of patterns, but signal patterns and locations randomly change *during* episode; (iii) Multi-class wide spectrum environment (most realistic): multiple signal classes with different behaviours (signal modulations), aperiodic, random activity and more bands (100 vs. 20). In (iii), observation and internal state includes *objectness* (is signal present) and *signal class score* (softmax score vector). Also added multi-class loss, metrics and rewards.



(a) SpecA environments



(b) SpecB1 environments

Figure 6: Comparison: hand-coded controllers expecting SpecA vs. ConvLSTM-DAN only trained in SpecA environments: (a) testing in SpecA; (b) testing in SpecB1. Hand-coded performance degrades significantly more than DAN when environments deviate from expected/training spec. Plot: Block IoU vs. time; 100 sampled test environments per spec.

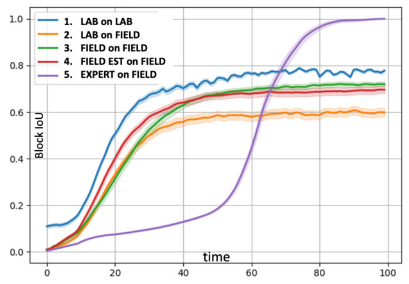
Table 2 shows the results of our approaches. Notice that InfoMax performs best in stationary environments, where the signal parameters vary a lot from episode to episode, but within the episode, they are stationary. Predictive-DAN is best at non-stationary environments, and RL methods (as opposed to non-RL based InfoMax) seem to perform best there. InfoMax-DAN seems to perform best at complex multi-class environments. It also shows that our designs can scale in frequency.

4.3 Experience Feedback (Field Spec Estimation)

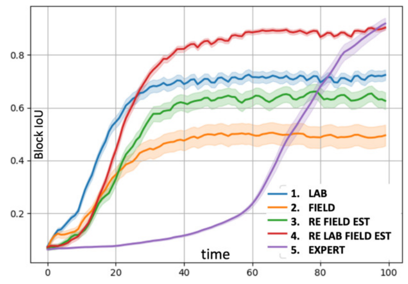
We now cover our results in leveraging experience feedback via field spec estimation and controller retraining using the estimated spec, while field state estimation is covered in Sec. 4.4. For these experiments, our Predictive-DAN is the ML-based controller used.
Estimating Spec from Feedback. In Fig. 7a we compare our approach (4. *Field Est on Field*) to a scan-and-dwell hand-coded control where field spec is within-distribution (5.*Expert on Field*) and also to our ML controller alternatively trained (1.-3.). The hand-coded controller takes a long time to reduce uncertainty (large search space). A DAN controller trained on lab alone cannot cope with field (2. *Lab on Field*). By learning estimated field spec *without* ground truth (GT), our approach has similar performance to training on field spec *with* GT (3.*Field on Field*, see also analogous 1.*Lab on Lab*).
Estimating Spec from Retraining. In Fig. 7b, we compare a controller (4. *Re Lab Field Est*) trained on Lab spec, then re-trained on a mix of estimated field spec (0.7 weight) + Lab spec (0.3 weight) to a scan-and-dwell hand-coded controller (5. *Expert*) and also to our ML controller alternatively trained (1.-3.). For this experiment, Field Spec and Lab Spec differ in number of signals (1 vs. 2, respectively) and width between signals (3 vs. 2), and we sample a mix of both specs for testing. Controllers trained on only one spec (1. *Lab*, 2. *Field*) cannot cope with environments from multiple specs. The controller retrained only on estimated field spec (3. *Re Field Est*) suffers from "forgetting" and cannot cope with environments from it's initial training spec. By retraining on both specs (weighted), the controller has the best performance.
Boostrapping. In Fig. 8, we consider a bootstrap loop with 3 iterations, where the ground-truth field spec parameters differ slightly from the previous iteration's (lab spec used is A and the three field specs are F1-F3 in Table1). The idea is to expose a controller to environments of variable complexity. The results for the last iteration are shown in Fig. 8, where we evaluate against *all* environment specs (lab + 3 field). The detection rate ($\approx 60\%$) of the controller with estimated specs (1. *Retrain Estimated*) is close to that of controller using GT specs (3. *Retrain Ground Truth*) $\approx 65\%$. Hence, training with GT loses advantage over training with estimated specs over multiple iterations as the agent is exposed to more and more types of environments. We can also see that re-training DAN controller leads to smooth adaptation and using only current estimated spec (2.*Cur Estimated*) results in poor performance.

4.4 Experience Feedback (Field State Estimation)

Instead of estimating the parameters of a field spec, which is then used to train a controller, we can perform *Field State Estimation*: do ML-based estimation of the full field/spectrum state (2. in Fig. 4) from partially observed field observations (1. in Fig. 4), and then use these reconstructed full-state episodes directly to retrain our ML controller, as described in Sec. 3.3.2. The ML-based estimation is done by a generator that is trained offline, separately from DAN. Once trained, it then generates one full-state episode from one partially observed field episode. In Table. 4, we compare the field state estimation approach to a controller exhaustively trained on an (anticipated) lab spec and one trained on the ideal training source: the field spec with complete ground truth information. We also study the performance after training with different numbers of reconstructed full-state episodes (25, 50, 100). Finally, we compare the approach with fine-tuning a controller in the field alone using partially observed prediction rewards—reward only what you observe in the band sampled and no error information using GT in other bands. For these experiments, we use *SpecC1* (Table 1) for the lab spec and *SpecC2* for the field spec. Throughout these experiments, evaluations of all training configurations are done on the same 100 field spec episodes (not used in training) and Predictive-DAN is the controller used.



(a)



(b)

Figure 7: Test results plotting timesteps vs. Block IoU comparing: (a) Estimating Spec from Feedback. (b) Estimating Spec from Retraining. (Block IoU of 5 steps.)

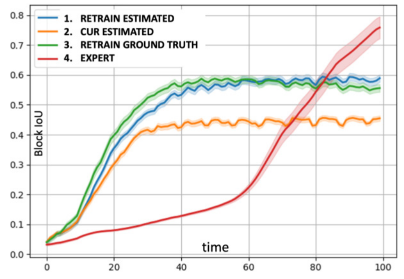


Figure 8: Comparison of DAN vs Expert controllers for the bootstrapping approach: performance on 3rd (last) iteration shown. (Block IoU of 5 steps.)

Training approach	Block IoU
1 Trained with lab spec only	0.58
2 Retrained w/ 25 estimated field state episodes	0.53
3 Retrained w/ 50 estimated field state episodes	0.70
4 Retrained w/ 100 estimated field state episodes	0.72
5 Training w/ field spec and full GT	0.75
6 Finetuned w/ 100 partially observed field episodes	0.43

Table 4: Experience feedback using *Field State Estimation* (rows 2-4), compared with training on (1) lab only, (5) field with full GT and (6) partially observed field episodes. Block IoU over last 33 steps.

The first row in Table. 4 corresponds to DAN trained with the lab spec, with no field experience and 1000 unique episodes. We can see that the performance is poor. The next three rows (2-4) correspond to DAN retrained with 25/50/100 estimated full-state field episodes mixed with 25/50/100 lab spec episodes, respectively. We can see that performance improves when the lab controller is retrained with more estimated field episodes. The fifth row in Table. 4 corresponds to training from the field spec with full ground truth and for 1000 unique episodes. This result roughly corresponds to the best possible performance of DAN given GT. As we can see, in this case, retraining DAN with 100 *estimated* field episodes (using no GT) has similar performance to training DAN with field GT and 1000 episodes.

The result of evaluating the DAN controller fine-tuned directly on 100 *partially observed field episodes* alone is shown in the last row (6) of Table. 4. For this case, we modified training for DAN accordingly to accommodate partially observed states, i.e., we have ground truth information only in the particular band sampled instead of all the bands previously—hence the reward obtained by M-network is limited to the prediction only in this band. We can see that its performance is inferior compared to training with full-state episodes, even when those complete states are estimated (row 4 of Table. 4). Partial observations in sparse, dynamic signal environments makes training difficult.

In summary, we show that retraining a DAN with full-state sequences estimated using ML-based generators from partially observed field observations is a viable and promising approach, while fine-tuning a DAN *directly* on partially observed field experiences produces poor results in general.

5 CONCLUSION AND FUTURE WORK

Novel RL-based approaches for information gain under partially observable, non-stationary and reward-sparse environments are presented and applied to the complex problem of RF spectrum monitoring, where the task is to identify signals with distinct behavioral patterns at variable frequencies. The proposed solution accounts for scalability and translation-invariance challenges of the signals in frequency-time space. We also propose information gain rewards to encourage exploration of unseen signals. Additionally, we developed two model-based RL approaches for the difficult task of retraining/fine-tuning a lab-trained controller using experience feedback from limited field deployment without ground-truth. Simulation results indicate that our approach outperforms previous RL and hand-coded solutions. We also show promising results for retraining our ML controller using limited field experience.

For future work, we plan on studying adversarial environments with competitive multi-agent learning and complex control surfaces with action representation learning. Also, we plan to investigate the use of GANs to generate training sequences from sparse field samples.

ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under Contract No. N66001-18-C-4044.

REFERENCES

[1] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. 2018. Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning. *CoRR* abs/1803.00101 (2018). arXiv:1803.00101 http://arxiv.org/abs/1803.00101

[2] Horacio Franco, Chris Cobo-Kroenke, Stephanie Welch, and Martin Graciarena. 2020. Wideband Spectral Monitoring Using Deep Learning. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning* (Linz, Austria) *(WiseML '20)*. Association for Computing Machinery, New York, NY, USA, 19–24. https://doi.org/10.1145/3395352.3402620

[3] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2020. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1lOTC4tDS

[4] Albert Haque, Alexandre Alahi, and Li Fei-Fei. 2016. Recurrent attention models for depth-based person identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1229–1238.

[5] Yuxiu Hua, Rongpeng Li, Zhifeng Zhao, Honggang Zhang, and Xianfu Chen. 2019. GAN-Based Deep Distributional Reinforcement Learning for Resource Management in Network Slicing. In *2019 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014217

[6] Jie Huang, Rongshun Juan, Randy Gomez, Keisuke Nakamura, Qixin Sha, Bo He, and Guangliang Li. 2021. GAN-Based Interactive Reinforcement Learning from Demonstration and Human Evaluative Feedback. *CoRR* abs/2104.06600 (2021). arXiv:2104.06600 https://arxiv.org/abs/2104.06600

[7] Michael R James and Satinder Singh. 2009. SarsaLandmark: an algorithm for learning in POMDPs with landmarks. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. 585–591.

[8] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[9] Gabriel Kalweit and Joschka Boedecker. 2017. Uncertainty-driven Imagination for Continuous Deep Reinforcement Learning. In *Proceedings of the 1st Annual Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 78)*, Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (Eds.). PMLR, 195–206. https://proceedings.mlr.press/v78/kalweit17a.html

[10] Sammie Katt, Frans A Oliehoek, and Christopher Amato. 2017. Learning in POMDPs with Monte Carlo tree search. In *International Conference on Machine Learning*. PMLR, 1819–1827.

[11] Merima Kulin, Tarik Kazaz, Ingrid Moerman, and Eli De Poorter. 2018. End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications. *IEEE Access* 6 (2018), 18484–18501. https://doi.org/10.1109/ACCESS.2018.2818794

[12] Lianjun Li, Lingjia Liu, Jianan Bai, Hao-Hsuan Chang, Hao Chen, Jonathan D. Ashdown, Jianzhong Zhang, and Yang Yi. 2020. Accelerating Model-Free Reinforcement Learning With Imperfect Model Knowledge in Dynamic Spectrum Access. *IEEE Internet of Things Journal* 7, 8 (2020), 7517–7528. https://doi.org/10.1109/JIOT.2020.2988268

[13] Gihan J. Mendis, Jin Wei, Ariuna Madanayake, and Soumyajit Mandal. 2019. Spectral Attention-Driven Intelligent Target Signal Identification on a Wideband Spectrum. In *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*. 1–6. https://doi.org/10.1109/CCAAW.2019.8904904

[14] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 2204–2212.

[15] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. 2020. Model-based Reinforcement Learning: A Survey. *ArXiv* abs/2006.16712 (2020).

[16] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*. PMLR, 2778–2787.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597 [cs.CV]

[18] Yash Satsangi, Sungsu Lim, Shimon Whiteson, Frans A. Oliehoek, and Martha White. 2020. Maximizing Information Gain in Partially Observable Environments via Prediction Rewards. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1215–1223.

[19] Yash Satsangi, Shimon Whiteson, Frans A Oliehoek, and Matthijs TJ Spaan. 2018. Exploiting submodular value functions for scaling up active perception. *Autonomous Robots* 42, 2 (2018), 209–233.

[20] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. Mach. Learn. Res.* 10 (dec 2009), 1633–1685.

[21] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. arXiv:1509.06461 [cs.LG]

[22] Leye Wang, Wenbin Liu, Daqing Zhang, Yasha Wang, En Wang, and Yongjian Yang. 2018. Cell Selection with Deep Reinforcement Learning in Sparse Mobile Crowdsensing. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 1543–1546. https://doi.org/10.1109/ICDCS.2018.00164

[23] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. arXiv:1511.06581 [cs.LG]

[24] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*. 802–810.

[25] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *ArXiv* abs/2009.07888 (2020).