RESEARCH ARTICLE

DeepAlloc: Deep Learning Approach to Spectrum Allocation in Shared Spectrum Systems

MOHAMMAD GHADERIBANEH, (Member, IEEE), **CAITAO ZHAN**, (Member, IEEE),
AND HIMANSHU GUPTA, (Member, IEEE)
Department of Computer Science, Stony Brook University, Stony Brook, NY 11790, USA

Corresponding author: Himanshu Gupta (hgupta@cs.stonybrook.edu)

ABSTRACT Shared spectrum systems facilitate spectrum allocation to unlicensed users without harming the licensed users; they offer great promise in optimizing spectrum utility, but their management (in particular, efficient spectrum allocation to unlicensed users) is challenging. To allocate spectrum efficiently to secondary users (SUs) in general scenarios, we fundamentally need to have knowledge of the signal path-loss function. In practice, however, even the best-known path-loss models have unsatisfactory accuracy, and conducting extensive surveys to gather path-loss values is infeasible. Thus, the current allocation methods are either (i) too conservative in preventing interference that they sacrifice performance, or (ii) are based on imperfect propagation models and/or spectrum sensing with insufficient spatial granularity. This leads to poor spectrum utilization, the fundamental objective of shared spectrum systems. In this work, we thus propose to *learn* the spectrum allocation function directly using supervised learning techniques. Such an approach has the potential to deliver near-optimal performance with sufficient and effective training data. In addition, it has the advantage of being viable even when certain information is unavailable; e.g., in settings where PUs' information is not available, we make use of a crowdsourced sensing architecture and use the spectrum sensor readings as features. In general, for spectrum allocation to a single SU, we develop a CNN-based approach (called DeepAlloc) and address various challenges that arise in our context; to handle multiple SU requests simultaneously, we extend our approach based on recurrent neural networks (RNNs). Via extensive large-scale simulation and a small testbed, we demonstrate the effectiveness of our developed techniques; in particular, we observe that our approach improves the accuracy of standard learning techniques and prior work by up to 60%.

INDEX TERMS Spectrum sharing, spectrum allocation, deep learning, convolutional neural networks.

I. INTRODUCTION

The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [4]. The research community has addressed this capacity crunch via the development of *shared spectrum paradigms*, wherein the spectrum is made available to unlicensed (Secondary) users as long as they do not interfere with the transmission of licensed incumbents, i.e., primary users (PUs). Effective management (and in particular, allocation) of spectrum in such shared spectrum systems is challenging, and several spectrum management architectures have been proposed over the years [7], [8], [26], [46]. A significant shortcoming of these architectures and methods is that spectrum allocation is done very conservatively to ensure correctness, or is based on imperfect propagation modeling [11], [13] or spectrum sensing with poor spatial granularity. This leads to poor spectrum utilization, the

The associate editor coordinating the review of this manuscript and approving it for publication was Ding Xu.

fundamental objective of the shared spectrum systems. In this paper, we develop a learning-based approach to efficient allocation of spectrum in such shared spectrum systems.

A. MOTIVATION AND OVERALL APPROACHS

In general, to allocate spectrum efficiently to secondary users, we fundamentally need to have knowledge of the signal path loss between largely arbitrary pair of points. In practice, however, even the best known path-loss models [13], [33] have unsatisfactory accuracy, and conducting extensive surveys to gather path-loss values are infeasible and, moreover, may not even reflect real-time channel conditions. To circumvent the above challenge, we propose to instead learn the spectrum allocation function directly using supervised learning techniques. In scenarios where primary-user (PU) parameters may not be available (e.g., navy radars in CBRS band [1]), we use a crowdsourced sensing architecture where we utilize relatively low-cost spectrum sensors independently deployed with a high granularity [10], [12], [24], [43]. A sensing architecture also enables spectrum allocation based on real-time channel conditions [16].

We propose to use supervised learning techniques to learn the Spectrum Allocation (SA) function with the input (features) being the primary-user parameters, spectrum sensor (SS) readings, and secondary user (SU) request parameters, and the output (label) being the maximum power that can be allocated to the SU without resulting in any harmful interference to the PUs' receivers. Based on the insight that the input to the SA function can be represented as an image and thus the SA function can be framed as an image regression problem, we develop a convolution neural network (CNN) model to learn the allocation function as CNNs have been most successful learning models for image regression and classification tasks. To develop an effective CNN-based learning architecture, we develop techniques to represent the inputs to the SA function as an image, design an efficient CNN architecture, and address associated challenges.

B. OUR CONTRIBUTIONS

We make the following contributions.

1) We motivate and propose using Convolution Neural Networks (CNNs) for efficient learning of the spectrum allocation function. In particular, we develop an efficient CNN architecture based on *pre-training* a deep model using many *auto-generated* samples/images, followed by training using samples gathered over the given region.

2) We develop a novel technique to represent the spectrum allocation function input (i.e., the location and transmission/received powers of primary users or spectrum sensors, and the request parameters of the secondary user) as an image; such an image representation is essential to effectively use a CNN-based learning model. In addition, we develop techniques to minimize false positive errors, handle multi-path effects, and

further improve accuracy via synthetic samples, in our context of a CNN-based learning approach.

3) To allocate spectrum simultaneously to multiple concurrent SUs, we develop a deep-learning architecture based on recurrent neural networks (RNNs) with inputs from our CNN-based architecture for single SUs.

4) We evaluate our techniques using large-scale simulations and an outdoor testbed and demonstrate the effectiveness of our developed techniques. We observe that our approach improves the accuracy of other approaches by up to 60%.
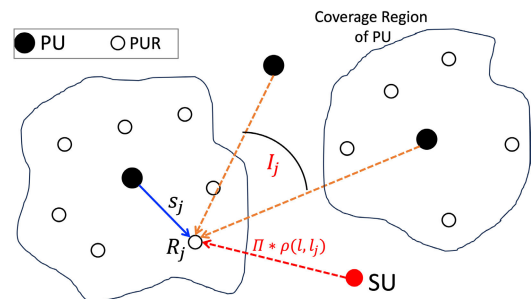
C. PAPER ORGANIZATION

The rest of the paper is organized as follows. In the following section, we develop our spectrum allocation model and setting, discuss related work, and give a high-level overview of our approach. In §III, we develop our CNN-based deep learning model and associated techniques for spectrum allocation. We discuss our simulation results in §IV, and end with concluding remarks in §VI.

II. MODEL, RELATED WORK, AND OUR APPROACH

Shared Spectrum and Entities: A shared spectrum system mainly consists of licensed primary users (PUs) and unlicensed secondary users (SUs) who make spectrum allocation requests to the centralized *spectrum manager*. A secondary user (SU) requests authorization to transmit with certain desired parameters (e.g., location, duration, frequency, transmission power). For a given SU request, the spectrum manager determines whether the SU's request can be granted based on whether its transmission under granted parameters would cause harmful interference to any of the intended receivers of a PUs signal, as discussed below. Additional entities in a shared spectrum system of relevance to our model are PU receivers and spectrum sensors, as described below.

PU Receivers (PURs). A way of modeling intended receivers of a PU could be to define a coverage region around PU wherein we wish to guarantee reception (see Fig. 1).

FIGURE 1. Eqn. (1) Illustration. The optimal power that can be allocated to an SU is such that, at each PUR (a PU's receiver) the signal-to-noise ratio is more than the desired ratio, β. Above, R_j is a certain PUR, I_j is the total interference at R_j from other PUs, s_j is the signal strength received at R_j from its PU, and $\Pi \cdot \rho(l, l_j)$ is the interference due to the SU at R_j where l and l_j are their respective locations.

In this work, as in [22] and [25], we model the intended receivers of a PU as a finite set of receiver nodes, which we denote as PURs. The PUR model is without loss of generality since, in general, the PURs can be distributed arbitrarily around a PU. The PUR model is also more computationally efficient than the coverage-region model in determining whether an SU's transmission causes harmful interference to PU's receivers. Later, we assume that the distribution method for PURs is "similar" across all PUs, to obviate the need to represent them explicitly in the SA's input image (see §III-A for more details).

Spectrum Allocation Objective: SU Transmission Power: The general spectrum allocation problem is to allocate optimal power to an SU's request across spatial, frequency, and temporal domains. We focus on the core function approximation problem, which is to determine the optimal power allocation to an SU for a given location, channel, and time instant—since frequency and temporal domains are essentially "orthogonal" dimensions of the problem and thus can be easily handled independently (as done in §III-F). We thus assume a single channel and instant for now, and discuss multiple channels and request duration in §III-F.

Determining Optimal Power Allocation: Consider a shared spectrum area with PUs deployed, and an SU request for transmission from a given location l. *If the PUs information and the path-loss function are known*, then the optimal power that can be allocated to SU without causing harmful interference to PUs can be computed as follows [16], [17], [22]. Let us denote the path loss function between a pair of given locations by $\rho(\ ,\ )$; thus, a signal transmitted at power t_i from location l_i yields a received power of $t_i \cdot \rho(l_i, l_j)$ at location l_j. Let the total interference from other PUs plus noise at R_j be I_j. To ensure that the signal-to-noise ratio at each R_j is more than the desired value, say β, the maximum power Π that can be allocated to the SU is:

$$\Pi \leq \min_j \frac{(s_j/\beta) - I_j}{\rho(l, l_j)},\qquad(1)$$

where $s_j = t_i \rho(l_i, l_j)$ is the signal strength received at R_j from its PU transmitting at power t_i from location l_i. See Fig. 1. Note that the above formulation is largely without loss of generality— as the path-loss function ρ can be arbitrary. Irrespective, *the fundamental techniques developed in our work are largely independent of the formulation or algorithm used to determine the optimal allocation power—* since learning models and techniques are solely based on training examples. In §IV, we use the above formulation to generate the training examples for the models.

Two Settings: `PU-Setting` *and* `SS-Setting`*:* The above formulation requires knowledge of PUs information as well as the path-loss functions. However, in most settings, neither of them may be available; in fact, this is one of the key motivations of our learning approach. In particular, PU parameters may not be available in military or government setting, e.g., in the CBRS 3.5GHz shared band [1] wherein the licensed users include Navy radar

systems. In light of the above, we consider two different settings in this paper, based on the availability of PU information/parameters.

`PU-Setting`. In this setting, the PUs' parameters are available—for determination of spectrum allocation. A PU's parameters include its location, transmit power, and its PURs' locations.

`SS-Setting`. In this setting, PUs' parameters may not be available, e.g., in military or government settings. In this case, to determine spectrum allocation, we make use of a crowdsourced sensing architecture where relatively low-cost spectrum sensors (SS) are deployed with a high granularity [10], [12], [24], [43]. In such a crowdsourced sensing architecture, allocation decision is based on SS parameters, which includes each sensor's location and received (aggregated) signal strength from the PUs (presumably representative of the PUs parameters). Allocation based on SSs parameters is implicitly based on *real-time* channel conditions, which is important for accurate and optimized spectrum allocation as the conditions affecting signal attenuation (e.g., air, rain, vehicular traffic) may change over time.

A. RELATED WORK

The spectrum allocation problem has been studied extensively (see [19], [46], [56] for a survey), especially in the context of shared spectrum systems. In a centralized SM architecture, it is generally assumed that the SM has complete knowledge of the PU parameters. Many prior works also assume a propagation model which, in conjunction with known PU parameters, allows spectrum allocation power to be computed via linear programming [46] or other simple techniques for common optimization objectives. However, in practice, PU parameters may not be available, e.g., in the CBRS (3550-3700 MHs, in the 3.5GHz) band [1] wherein the licensed users include Navy radar systems. As most propagation models have unsatisfactory accuracy, spectrum allocation must be done overly conservatively for correctness. In particular, in TV white spaces spectrum (54-698 MHz) [2], spectrum allocation is done based on a database with TV channel availability at each location; in essence, the SU is allowed to transmit with a certain power if the signal received at its location is below a low threshold. Such a **listen-before-talk** [30], [45] allocation strategy, in general, can be very conservative due to a combination of reasons: (i) the observed signal is actually an aggregate over all PUs in the same band, (ii) path-loss need not be symmetric, (iii) SUs may want to transmit at a much lower power than the PUs, and (iv) PURs may be a distance away from the PUs. In static systems, a database approach [1] can also be used by pre-computing the SA function and storing in the databases; however, such an approach doesn't work in any dynamic situation, e.g., SUs at the tertiary level, PUs changing powers, path-loss changes due to real-time conditions. In a closely related work, [22] has developed an interpolation-based spectrum allocation scheme

that works by first estimating the desired path-loss values based on signal strength readings from deployed spectrum sensors. In particular, they use inverse-distance weighted and Ordinary Kriging interpolation schemes to estimate the path-loss values. We compare our approach with theirs in §IV-§V.

1) MACHINE LEARNING (ML) BASED APPROACHES

To the best of our knowledge, there have been no prior works that have used supervised learning to directly learn the SA function, especially as defined here. The closest work is [5], which uses supervised learning to analyze spectrum *occupancy* based on the sensed signal at the SU. Deep learning models have recently been used (see [41] for a survey) to learn radio-propagation models and prediction; e.g., using CNNs [54], for large-scale fading in 5G cellular networks [39], using SegNet encoder-decoder model [38], using SVMs [47]. Path-loss models can then be used to allocate spectrum using Eqn. 1—however, Eqn. 1 requires knowledge of PU parameters, which may not be available. Moreover, the path-loss function fundamentally encodes more information than the SA function,[1] and thus, would likely require much more training.

Reinforcement Learning (RL) Approaches. Reinforcement learning (RL) models have been used in slightly different spectrum allocation settings wherein multiple agents are involved or competing with each other for spectrum resource and they undertake a sequence of actions. E.g., [21], [34], [37], [55], and [14] have applied RL techniques for power control in multi-agent cellular networks, wherein the agents interact with each other and cell towers to determine power allocation. In addition, RL-based spectrum allocation works in radio networks (see [35], [50] for surveys) have largely focused on channel assignment; in contrast, our work is focused on power allocation. Zhang et al. [53] propose a deep RL algorithm where a deep neural network is used to help SUs obtain information about PUs' power policies.

2) MULTIPLE CHANNELS AND OTHER OBJECTIVES

In this paper, we implicitly assume a single channel for the most part. Spectrum allocation for multiple channels can be done by using single-channel techniques independently for each channel and then selecting one of the available channels based on some criteria (see §III-F). For example, Wang et al. [48] picks a channel that maximizes the aggregate data rate of SUs, and Li et al. [31] picks a channel that allows for minimum transmission power for a desired SU data rate. Other works have addressed spectrum allocation with other optimization objectives. e.g., researchers have considered throughput maximization as an objective [40], [49] under various constraints such as maximum allocated power [31], given QoS requirements [29], etc. Fairness

and energy efficiency are some other criteria considered [9], [20], [51].

B. OUR LEARNING APPROACH

1) MOTIVATION FOR LEARNING SA FUNCTION

Our goal is to allocate spectrum efficiently to SUs in general settings, e.g., when PU parameters may not be available. To motivate and justify our learning based approach, we make the following remarks. <u>First</u>, we note that to allocate spectrum near-optimally to secondary users in general scenarios, we fundamentally need to have knowledge of the signal path-loss function. In practice, however, even the best known path-loss models [13], [33] have unsatisfactory accuracy, and conducting extensive surveys to gather path-loss values is infeasible and moreover, may not even reflect real-time channel conditions. In absence of knowledge of a path-loss function, to allocate spectrum efficiently, we propose to just learn the spectrum allocation function directly using supervised learning techniques. <u>Second</u>, in our context, an unsupervised approach is meaningless as unlabelled samples have minimal information (actually, zero information in the `PU-Setting`), and as explained in §III, a reinforcement-learning approach is also not suitable for our setting. <u>Third</u>, learning the path-loss function first and then using Eqn. 1 to allocate spectrum is certainly a feasible approach – but, since the path-loss function fundamentally encodes more information than the SA function, it would likely require much more training (note that the SA function depends only on the most restrictive of the PURs). <u>Finally</u>, non-trivial parameters such as weather, terrain and obstacles, PU transmitters being directional, etc., can be relatively easily incorporated in a learning approach (see §III-F), while they would require more sophisticated modelling techniques and algorithms to be incorporated in non-learning approaches.

Based on the above observations and insights, we propose to learn the SA function directly from training examples. In general, our goal is to learn the SA function accurately with minimal training. Below, we discuss the inputs/features of our learning models, and the gathering/generation of training samples.

2) INPUTS/FEATURES OF THE SA FUNCTION $\mathcal{A}$

For most of the discussion in this paper, we focus on spectrum allocation to a *single/first* SU in a given area; multiple or subsequent SUs can be handled similarly as discussed later in §III-F. In the simplest of settings, the PUs' information/parameters (location and transmit power) do not change over time, in which case the SA function can be simply represented as a function of just the SU's location, as the fixed PU parameters will be automatically captured within the learned model. In this work, we focus on the more general settings wherein the PUs' information (location and power) may change across training and evaluation samples. For our two settings, viz., `PU-Setting` and `SS-Setting`, inputs/features of our SA function are as follows.

---

[1]Note that the complete path-loss function is sufficient to estimate the SA function, but not vice-versa. Estimating SA function from path-loss is straightforward in the `PU-Setting` using Eqn. 1, while in `SS-Setting` one can first estimate the PU parameters with reasonable accuracy from path-loss function and SSs readings.

1) `PU-Setting`. In this setting, the PUs' parameters are available for spectrum allocation determination. Here, the inputs to the SA function are PU parameters $\mathcal{R}$ and location l of the requesting SU, and thus, the SA function can be represented as a real-value function $\mathcal{A}(\mathcal{R}, l)$.

2) `SS-Setting`. In this setting, PUs' parameters may not be available, and, as mentioned before, we use spectrum sensors (SSs) to gather received power and use sensors' readings to determine spectrum power allocation. Here, the inputs to the SA function are (i) SSs parameters $\mathcal{S}$, and (ii) location l of the requesting SU. Thus, the SA function is can be represented as $\mathcal{A}(\mathcal{S}, l)$. For each SS, its parameters may include its location and aggregate received power from the PUs, and in general, may also include the mean and variance of the Gaussian distribution of the received power.

3) GATHERING AND LABELING TRAINING SAMPLES

Note that gathering a training sample for $\mathcal{A}$ entails gathering feature values and determining its "label"—in our context, for a given feature vector (SU location and PUs/SSs parameters), the label is the maximum power that can be allocated to the SU without causing harmful wireless interference at *any* of the PURs. In the `PU-Setting`, the features are the available PU parameters. In the `SS-Setting`, for gathering training samples, we need to deploy SSs and gather received powers (see §V) for collecting sample features. In either setting, to label the sample, we need to estimate the maximum allowable power at a given SU's location; this entails simulating PURs and determining the maximum SU transmission power that allows PURs to receive the PU signal (i.e., ensures that the signal-to-noise ratio is above a desired constant). To estimate the maximum SU power allowed, we can do a binary search on SU power, as done in our tested setup described in §V. To circumvent the collection of noisy samples, we can throw out samples that cannot be reproduced. We acknowledge that the training process can incur a substantial cost, but can be automated using drones for entities. More importantly, training is done only one-time, and thus, some amount of training cost is tolerable.

III. CNN-BASED DEEP LEARNING APPROACH

In this section, we motivate the convolutional neural network's (CNN) suitability for our context, and design an efficient CNN architecture and associated techniques for our problem.

Motivation for Using CNNs for SA Function: We observe that SA function can be looked upon as an image regression function, with the inputs to the SA function (SU location, PUs/SSs parameters, etc.) represented as an image and the regression value representing the optimal allocation power. Framing SA function as an image regression function allows us to leverage known advanced image regression

models. In particular, CNNs have been very successful in image classification or regression tasks or in capturing patterns/objects in images, because CNNs are able to exploit the spatial structure in images via the use of learnable spatially-localized filters (or kernels) in its convolution layers. In our context, the spatial nature of the SA problem (i.e., entities deployed over a geographic area) means that the input to the SA problem can be represented as a 2D image with sufficient accuracy. Such an input representation allows us to use CNNs to learn the SA function effectively. We corroborate our above intuition about the suitability of CNNs to our problem via extensive evaluations in §IV. For the general case of multiple SUs, we augment our CNN model with Recurrent-Neural Networks (RNNs) in §III-F.

Radio propagation modeling and prediction using CNNs cnn-4 overview cnn-3 large-scale fading in 5G cnn-2

Challenges. However, there are significant challenges that need to be addressed, to make CNN a viable and efficient approach to learn the SA function. These include the pre-processing of samples into "images" to feed as input to a CNN model, creating an efficient CNN architecture, ensuring minimal false positives, handling multi-path fading effects, minimizing training costs, etc. We discuss these challenges in the following subsections.
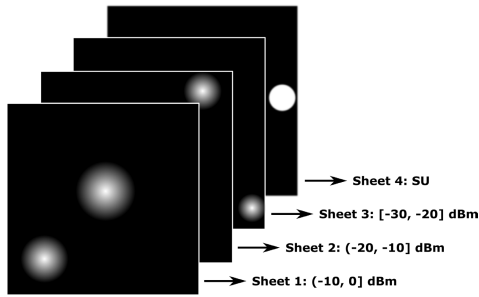
Other Machine Learning Models: We believe that CNNs are best suited to model the (single SU) SA function. However, other machine learning models can also be used to learn the SA function—in particular, we also evaluate neural networks (NNs) and space-vector machines (SVMs) approaches in §IV-A.

However, we note that reinforcement learning (RL) approach is not suitable for learning the spectrum power allocation function, as defined in this work, for the following reasons. Learning our SA function is fundamentally a supervised learning problem—since our learning goal is to approximate a function using *labeled* training examples (which can be gathered, as discussed before). In contrast, in RL settings, an agent learns a policy about which action to take on each state so as to maximize the cumulative reward; the learning of such a policy is driven by (i) the rewards given by the environment based on the current/next state(s) and/or the action taken, and (ii) the Markov-decision process (MDP) over the system states and actions that represent how the system transitions through states potentially influenced by actions. As there is no underlying MDP governing our spectrum allocation setting, the RL approach is not suitable for our context.[2]

A. `SH-ALLOC`: *SHALLOW CNN MODEL*

In this subsection, we discuss our basic CNN architecture and approach, which we refer to as `SH-Alloc`, as it has

---

[2]If we use the RL technique in our setting by considering actions as power allocations, we'll need to provide training examples for *every* possible system state, due to the lack of an underlying MDP, making the approach infeasible. Note that in a setting with no underlying MDP, the RL approach learns the policy independently for each state.

FIGURE 2. Image representation of a sample for input to the `DeepAlloc` model. Here, there are four PUs (2 in the first sheet, and 1 in the other two sheets) and one SU (in the fourth sheet). Each sheet for the PUs corresponds to a range of PU's transmit power; e.g., PUs whose transmit power is in the range −10 to 0 dBm are placed in the first sheet. When representing SSs in `SS-Setting`, we place SSs in the sheets based on their locations.

a small number of layers. In the next subsection, we will extend this approach to the `DeepAlloc` approach that uses a much deeper CNN architecture with a larger number of layers. We start with discussing our strategy to pre-process training samples into images.

1) PRE-PROCESSING TRAINING SAMPLES TO IMAGES

The first challenge in applying CNNs to our context effectively is to transform each training sample to an "image" for input to the CNN model, in a way that it is most conducive to efficient learning. Representing the entities (PUs, SSs, SUs) as objects in a 2D image is a natural choice. In particular, we could represent each entity type with a different color or shape, or more specifically, represent each entity by a disk of a certain color with a radius based on the transmit/received power. Our choice of image representation is tantamount to *feature engineering* [6], and can have a significant impact on the training cost. Below, we discuss the choice made in our model design. First, we assume that the PURs are distributed similarly across all PUs; more formally, we assume that the distribution of PURs around its PU P is a function of P's parameters (location and power). Under the above assumption, we do not need to represent PURs of any PU in the input image—as the distribution of PURs can be learned by the model from each PU's parameters. E.g., if for each PU P, its PURs are distributed uniformly within a fixed radius around P (or within a radius proportional to P's power), then we don't need to represent PURs in the input image.

`PU-Setting`: Representing PUs in Multiple Image "Sheets." Note that just using shapes or colors for different entities is insufficient, as we also need to represent the transmit/received powers. Just using radius to represent powers is not viable either, as we may start getting intersections between shapes. Thus, we compose the input image of a certain number of "sheets" (see Figure 2). Then, we divide the expected transmit-power range into ranges of about 5-10dBm and assign each PU to the appropriate image sheet depending on its transmit power. Within each sheet, we then use disks to represent each PU, with the brightness of the center pixel as well as the radius of the disk proportional

to the transmit power. In addition, to give more importance to the center (which represents the true location) and to suggest signal attenuation away from the center, we decrease the brightness of the pixels away from the center in a logarithmic manner. If there is an intersection between two objects in the same sheet, we aggregate the intensity of the common pixels. Unlike normal images, which are composed of three sheets corresponding to red, green, and blue colors, we may use more than three sheets.

`SS-Setting`: Representing Spectrum Sensors' (SSs) Readings. SSs can be represented similarly to PUs with the size of their disk proportional to the *received* (rather than transmit) power. However, unlike PUs, we place SSs among the sheets based on their location rather than received powers, e.g., SSs from certain grids were always placed on the first sheet irrespective of their received-power readings. In our evaluations, we observed that placing SSs over sheets based on locations improved the performance of our models significantly compared to placing the SSs based on received powers.
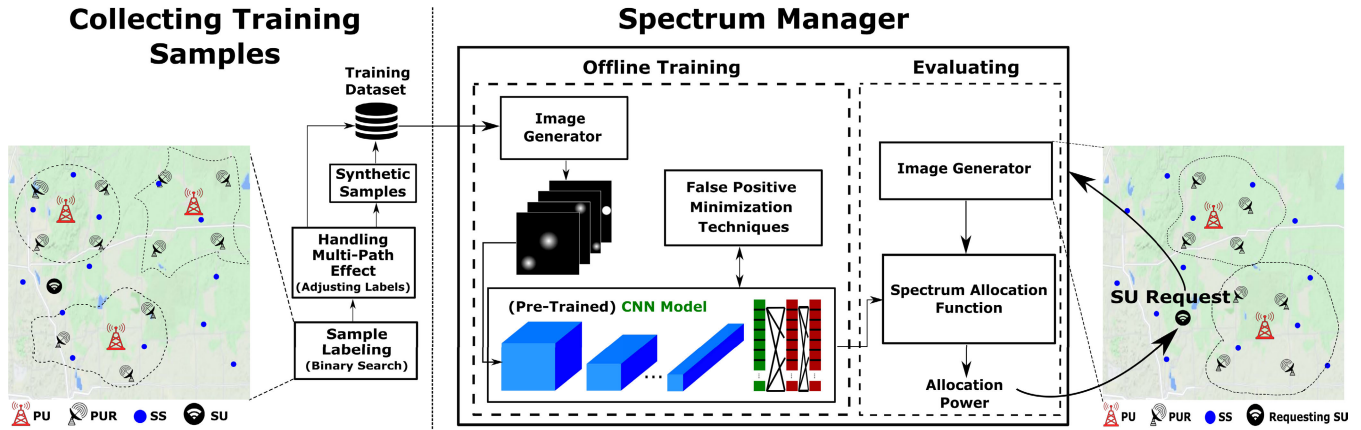
Representing SUs. We use a separate sheet to represent SU(s), instead of representing them with a different shape/color, to facilitate potentially more efficient training. `SH-Alloc` **CNN Model Architecture.** To design a CNN architecture to learn the SA function efficiently, we need to carefully determine the various model parameters. For our `SH-Alloc` model, we choose these parameters as follows. **(i)** Number of convolution layers in a CNN model plays an important role in training cost as well as model accuracy. In general, a deeper model (i.e., with more number of convolution layers) performs better than a shallow one but incurs more training costs. In our context, only a few thousand training samples are feasible to gather, we use 5 convolution layers, and 3 fully-connected layers as in a neural-network architecture. **(ii)** *Filter Size:* To be able to "detect" small-radius PUs of small power values, we use a 3×3 filter in the convolution layers. The number of filters is small for the initial layers, and then increases with the "depth" ; this facilitates the detection of more and more complicated features in the deeper layers. **(iii)** *Activation Function:* In our context, as the final output of the model is a real number, we chose a linear activation function for the last network layer with only one neuron. For other layers, we use Rectified Linear Units (ReLU) [36] activation function as its non-linearity enables the model to learn a more complex function faster.
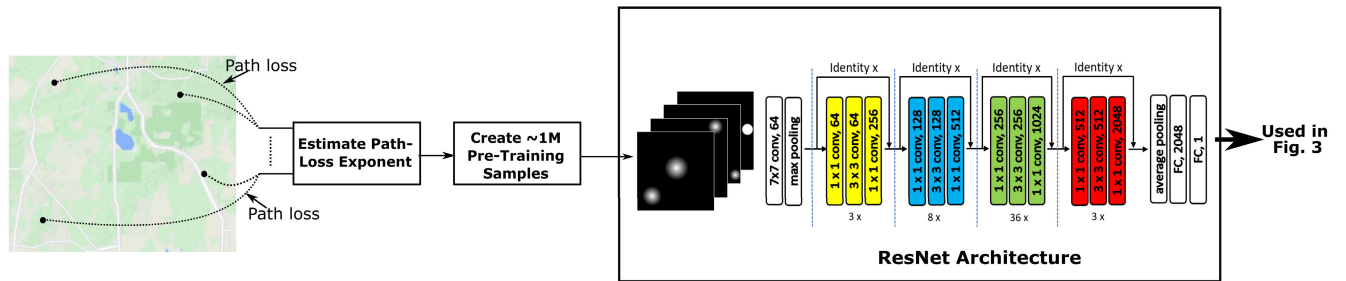
The overall CNN-based spectrum allocation system is shown in Fig. 3, including components discussed in the following subsections.

B. `DEEPALLOC`: PRE-TRAINED DEEP CNN MODEL

It is well understood that deeper neural networks, i.e., neural networks with more layers, and in particular, deeper CNN models can yield much better performance with their ability to learn more complex functions, with sufficient training data.

FIGURE 3. CNN-based Spectrum Allocation System. For the CNN Model component, `SH-Alloc` uses a shallow CNN model without any pre-training, while `DeepAlloc` uses a deep pre-trained CNN model obtained from Fig. 4. In both cases, after any pre-training, the (field) training samples are gathered, converted into images, and then used to (further) train the CNN model. The learned model is then used to allocate spectrum to requesting SUs.

FIGURE 4. `DeepAlloc` Pre-Training Process. Here, we generate a large number ($\approx$ 1M) of images assuming a log-normal propagation model based on an estimate path-loss exponent, and pre-train a deep CNN model (we used ResNet) using these generated images. The pre-trained model is then used in Fig. 3 for further training using a smaller number of field training samples.

However, there are two major challenges with using deep models. First, as the number of layers increase, so does the number of parameters to learn—which in turn requires a much larger number of training samples. Second, a deeper model is more likely to learn a function that overfits the training data, if the training data is not sufficient. The perfect solution is to use a deep network but also have a sufficiently large training set. However, in our context, it is infeasible to gather more than a few thousand samples as the deployment and/or labeling costs are high, while a very deep (e.g., with 20-100 layers) network may require close to a million samples to train properly [15], [23], [44].

1) USING PRE-TRAINED MODELS

Our approach to address the above challenge is to use a *pre-trained* deep model, i.e., a deep model that has already been trained with readily-available images, which may not represent spectrum allocation samples. Such a strategy has been often used in computer vision applications with great success. Similarly, we could also use pre-trained deep well-known models such as VGG [44], ResNet [23], Xception [15] which have been trained with around 1 million daily-life images, and then train them further with a few thousand spectrum allocation training samples, as in the previous subsection.

`DeepAlloc` Approach. One unique aspect of our context is that the pre-training samples or images can be easily synthesized based on an assumed propagation model. Thus, to obtain better performance than the above models are pre-trained with daily-life images, we can pre-train a deep architecture (we use ResNet [23][3]) with generated spectrum allocation images based on an assumed log-normal propagation model with an ''appropriate'' path-loss exponent. (Results in Fig. 10(b) of §IV-A validate the above pre-training approach.) The path-loss exponent α can be derived by gathering a few (say 100-200) path-loss samples, and determining the best α that minimizes the error between the actual samples and those from the log-normal model. See Fig. 4. Then, as before, we train such a pre-trained model further with a few thousands of SA training samples as shown in Fig. 3. We refer to this overall approach as `DeepAlloc`.

C. MINIMIZING FALSE POSITIVE ERROR

Note that we should rarely allocate power higher than that determined by Eqn. 1, as it would cause harmful interference to some PURs. Ideally, we would like to minimize such cases, which we call *false positives*, drastically—perhaps,

---

[3]We evaluated VGG-based `DeepAlloc` too but it was easily outperformed by the ResNet-based `DeepAlloc` architecture especially in the `SS-Setting`.

at the cost of higher number of (and/or errors in) the false negative cases. We address this issue by a combination of two strategies. First, we choose the hyper-parameters of the CNN model in a way that minimizes the number of false positives; the methodology here is largely trial and error, due to a lack of techniques to systematically search for hyper-parameter values. Second, we use asymmetry in the training loss function as follows. Essentially, we change the training loss function $J(\theta)$ such that false positive samples (SU requests that get higher power than the maximum) are penalized more drastically than the other requests. More formally, we define the training loss function as:
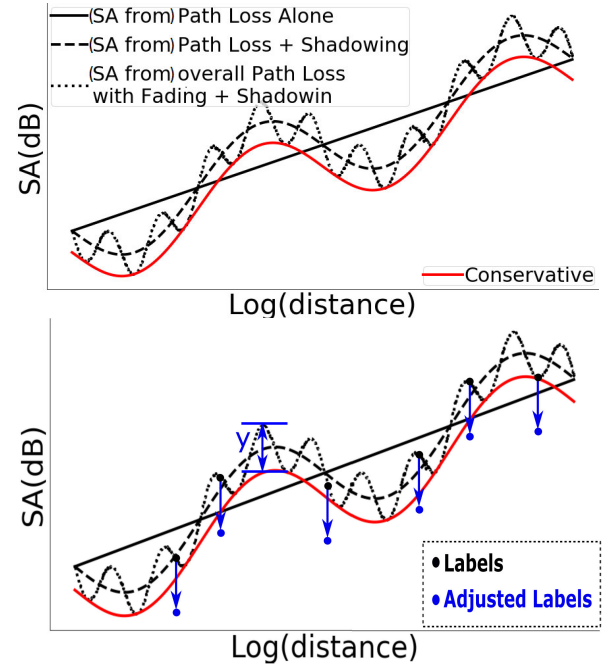
$$J(\theta) = \frac{1}{m} \left(\alpha_{FP} \sum_{\hat{y}_i > y_i} l(\hat{y}_i, y_i, \theta) + \alpha_{FN} \sum_{\hat{y}_i \leq y_i} l(\hat{y}_i, y_i, \theta) \right)$$

(2)

Above, α_{FP} and α_{FN} are coefficient weights for the false positives and false negatives respectively, y_i is the "ground truth", $\hat{y}_i$ is the predicted value, θ is the internal set of the parameters being learned, and $l(.)$ defines the error for a single sample. To minimize false positives, we can choose a much higher α_{FP} than α_{FN}.

D. HANDLING MULTI-PATH EFFECT

Wireless signal attenuation is a result of three mutually independent, multiplicative propagation phenomena [42], viz., large-scale signal attenuation, medium-scale shadowing, and small-scale multipath fading. The shadowing effect can cause overall path loss to vary over 10s to 100s of meters, and the fading effect can cause an effect over a few wavelengths. See Fig. 5 (a). The overall effect of these phenomena in our context is that learning the SA function could require a much higher number of training samples—to sufficiently and accurately capture the small to medium scale fluctuations. This challenge is fortunately mitigated by the fact that the small-scale fluctuations in the path-loss function are unlikely to fully manifest in the SA function, as the SA function depends only on the most restrictive of the SU-PUR path losses.

In our context, one way to address the above challenge is to learn instead of a more "conservative" SA function that may be easier to learn but may allocate sub-optimal power. E.g., the conservative SA may be based on a *lower-bound* approximation of the path-loss function. See the red curve in Fig. 5 (a). We note that the technique in the previous section is also driven towards learning a more conservative function via an appropriate training loss function. However, the model inaccuracy resulting from the multi-path effect is fundamentally due to a lack of sufficient training samples needed to learn a function with high spatial resolution/variability; thus, we can't fix the model inaccuracy due to the multi-path effect by merely changing the loss function, model, or its training process. Our strategy to learn a more conservative function that addresses the multi-path

FIGURE 5. (a) Spectrum allocation function, when there is a single PU and a single SU, due to path loss with shadowing and multi-path fading effects. Note that the path-loss function has a similar trend. The red plot is a conservative spectrum allocation, based on a similar conservative path-loss function. (b) Modifying labels of training samples to drive the model towards learning a simpler and more conservative spectrum allocation function.

effect challenge is to actually modify the labels of the training samples appropriately, as discussed below.

1) LEARNING A CONSERVATIVE SA FUNCTION BY MODIFYING TRAINING SAMPLE LABELS

To ensure few false positives in the face of the small-scale multi-path the fading effect, we modify the training samples by lowering their labels from the optimal allocated power to smaller values so that the modified training samples represent a "conservative" spectrum allocation function without the small-scale effects. In particular, let's consider a training sample–corresponding to a certain set of PUs with given transmit powers, PURs, and SU S_i with allocated power Π which satisfies Eqn. 1.[4] (Our discussion in this subsection applies to both settings, viz., `PU-Setting` and `SS-Setting`.) Let l_i be the location of the SU S_i. Now, let PUR R_j be the PUR that results in S_i's allocated power, i.e., let $j = \arg\min_j \frac{\tau_j - I_j}{\rho(l_i, l_j)}$ from Eqn. 1. In the following discussion, we vary l_i while keeping everything else constant–thus, we can look at j, R_j, and allocated power Π as functions of l_i. Now, in a sufficiently small neighborhood N of S_i, the PUR $R_j(l_i)$ as computed above and thus $\Pi(l_i)$ must remain fixed for $l_i \in N$. Moreover, $\Pi(l_i)$ is also inversely proportional to $\rho(l_i, l_j)$ with l_j fixed, and more importantly, variation of $\Pi(l_i)$ within N is largely dominated by the small-scale effect of the

---

[4]However, note that Π is not computed *using* Eqn. 1, as many terms in it are unavailable.

path-loss function. Let $y = \max_{l_i \in N} \Pi(l_i) - \min_{l_i \in N} \Pi(l_i)$, which in some sense is an estimate of the "amplitude" of the small-scale effect. We now use the value of y to lower the allocated power label of the training samples. See Fig. 5(b). The modified samples essentially correspond to a conservative spectrum allocation function, without the small-scale effect. Note that the value y depends on the small-scale effect of the terrain. If we assume the effect to be uniform across the given region, then the same value of y can be used across all the training samples—else, we compute y in each subarea.

E. SYNTHETIC SAMPLES TO IMPROVE PERFORMANCE

One way to improve performance, i.e., to aid the model in extracting the most information from the given training samples, is to create additional *synthetic* samples from the training samples. In effect, the new synthetic samples incorporate the domain knowledge used in creating them. In general, from a given sample $\{X, y\}$ where X is the set of features (PU parameters or SS readings, and the requesting SU's location) and y is the label (allocated power), we create synthetic samples of the type $\{X', y\}$ where X' is another set of features that yields (approximately) the same label y. Below, we discuss the generation of synthetic samples for our two settings, and evaluate the improvement from these strategies in §IV-A.

1) PU-SETTING

Consider a sample $\{X, y\}$ where X includes the PU parameters. For this sample, let P be the PU for the PUR R_j that determines the SU's optimal power allocation, as in the previous subsection. Let $\mathcal{P}$ be a set of PUs that are sufficiently far away from P. Note that P is determined when labeling the given sample, and $\mathcal{P}$ can be determined from P and the available PU parameters. Now, note that decreasing the power of PUs in $\mathcal{P}$ should not change the optimal power allocated to SU. Thus, we can create additional synthetic samples by considering X' which differs from X in that the transmit powers of PUs in $\mathcal{P}$ is lower than that in X.

2) SS-SETTING

In the setting, wherein the features are composed of sensor readings, we generate synthetic samples by determining sensor readings at additional locations via interpolation techniques. In particular, based on the log-distance-like behavior of signal attenuation, we use a slight modification of the traditional IDW technique. More formally, for a known set of sensor readings (p_i, l_i) where p_i is the received power at location l_i, the interpolated value q at a new location l is given by: $q = \frac{\sum_i w_i p_i}{\sum_i w_i}$, where w_i is the weight defined as $w_i = \frac{1}{\log_{10}(d(l_i, l))}$. Based on the above interpolation scheme, for a given sample (X, y) where X is the set of real sensor readings, we can create synthetic samples of the type (X', y) where X' consists of some readings from X and some interpolated readings.

3) ROTATED IMAGES

In addition to the above, we also synthesize additional samples by just "rotating" the images of the original samples; such a strategy is bound to be useful in regions where the propagation model is largely dependent on the distance. In our evaluations, we rotate the given images by 90 or 180 degrees.

F. MULTIPLE SUS

Till now, we have only considered spectrum allocation to the initial single SU. Handling *subsequent* SUs in the same manner as the initial SU above requires allocated power to the active SUs. We start with discussing how to handle multiple SUs *one at a time*, and then discuss handling multiple SUs simultaneously.

1) HANDLING SUS ONE AT A TIME

To handle subsequent SUs, we can augment the list of features to include the active SUs' parameters and train the model accordingly. This approach is viable, especially in the context of our `DeepAlloc` model, since active SUs with their parameters (location and allocated power) can easily be represented in the single input sheet dedicated to SUs. The only change we need to make is that we must now represent SU's allocated power too; we can represent them with a disk similar to PUs, i.e., with the center's brightness and radius proportional to the allocated power. If multiple SU requests arrive *together*, then to use the above approach, we need to decide on an order in which to handle the requests. One approach is to handle them in a greedy order as follows; we call this approach `DeepAlloc-Greedy`. We divide the whole area into non-overlapping subareas and assign each subarea X a weighted score equal to the total aggregated power transmitting from all the PUs and active SUs in X or its adjoining cells. Then, we pick the SU requests in the ascending order of the weight of the subarea they belong to.

2) HANDLING MULTIPLE SUS SIMULTANEOUSLY

Handling SUs one at a time, though easier and simpler, can lead to unfair and/or far-from-optimal (in terms of say, *total* power allocated) allocation. Thus, we now discuss how to extend our deep-learning models to facilitate simultaneous allocation of powers to multiple SUs. Recall that our `DeepAlloc` model is trained to predict the maximum power that can be allocated to a single SU. To allocate powers to multiple SUs $\{S_1, S_2, \ldots, S_n\}$ simultaneously, we need to essentially learn a function that maps the list of SUs' locations $\langle l_1, l_2, \ldots, l_n \rangle$ to the list of simultaneous allocated powers $\langle Q_1, Q_2, \ldots, Q_n \rangle$. To learn the above function, we use two different learning approaches.
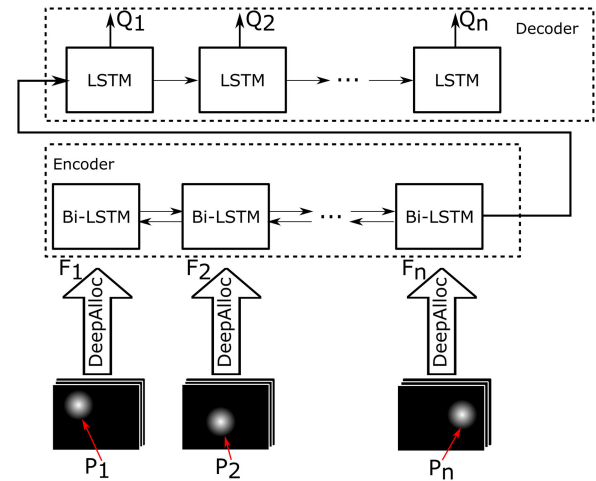
(1) `DeepAlloc-NN`. The simplest approach is to use a traditional neural-network (NN), with inputs as the list of SUs' locations $\langle l_1, l_2, \ldots, l_n \rangle$ and other available PU or SS parameters, and the output as the list of allocated simultaneous powers $\langle Q_1, Q_2, \ldots, Q_n \rangle$. To leverage our developed

single-SU model `DeepAlloc` and facilitate more efficient training, instead of passing the PU or SS parameters, we pass the set of "independently allocated" powers $\{P_1, P_2, \ldots, P_n\}$ where P_i is the power allocated to S_i assuming no other SU $S_j(j \neq i)$ is active. Thus, the input to the NN is the list $\langle (P_1, l_1), (P_2, l_2), \ldots, (P_n, l_n) \rangle$, with the output being the list of simultaneous power allocations $\langle Q_1, Q_2, \ldots, Q_n \rangle$. The NN is trained using training samples with labels determined as described below. In our simulations, we used a 4-layer NN with 128 neurons in each layer and a dropout of 80% before the last layer.

(2) `DeepAlloc-RNN`. As mentioned above, we need to learn a model that maps a list/sequence of SUs' locations to a list/sequence of allocated powers. In recent years, the *Seq2Seq* models have achieved a lot of success in learning tasks that involve mapping a sequence of words to another sequence of words, e.g, in machine translation, text summarization, image captioning, etc. In particular, Google Translate uses *Seq2Seq* model to translate a paragraph from one language to another. The *Seq2Seq* model comprises of an encoder to encode the input sequence, and a decoder to map the encoded data to a target sequence. The encoder essentially creates a sequence of hidden state vectors each of which depends upon the previous input and state vector. Similarly, the decoder uses the created hidden state vectors with appropriate attention/weighting schemes to generate a target sequence.

As our multi-SU spectrum allocation problem entails mapping a sequence of SUs' information to SUs' power allocations, we develop a *Seq2Seq* model with appropriate inputs, outputs, and internal components as follows. For our purposes, we propose to use an RNN-based Seq2Seq model, which we call `DeepAlloc-RNN`. In particular, we implement the encoder using a bi-directional long short-term memory (LSTM) model and the decoder as a uni-directional LSTM. See Fig. 6. We use bi-directional LSTM as the encoder to facilitate interpretation of the input set of SUs as *unordered* to signify *simultaneous* allocation. In addition, rather than passing $\langle (P_1, l_1), (P_2, l_2), \ldots, (P_n, l_n) \rangle$ as the input to the RNN model as we did in the above NN approach, we pass the set of feature-sets $\{F_i\}$ as input where F_i is as follows (see below for the motivation). The feature-set F_i (for a given S_i) is the output of the average-polling layer (Fig. 4) of the `DeepAlloc` model, when the input (represented in sheets) is SU S_i's location l_i and independent power P_i and given PUs/SSs parameters. See Fig. 6. The above idea of passing F_i's to the RNN allows us to encode more information in the input to the RNN, rather than just the power P_i's. The final output of the decoder represents the powers allocated to the SUs together.

Determining Training Sample Labels (`Binary-Alloc`). To train the above models, we need a method to label the training samples; in particular, we need to design an algorithm to determine the optimal powers to allocate to multiple concurrent SUs. Our implicit performance metric is the *total* power allocated to the given set of SUs. Here, in the training

FIGURE 6. RNN-based Architecture for Simultaneous Allocation to Concurrent SUs. The input to the encoder is the convolution output F_i of `DeepAlloc` for each SU S_i, and the output of the model is the SU power allocations $\{Q_i\}$. The input to `DeepAlloc` for each S_i is a set of sheets representing S_i's location l_i and independent power P_i and PUs/SSs information.

phase, even though we can assume knowing whether a certain power allocation to multiple SUs causes interference to any PUR, to determine optimal power allocation to multiple SUs is still non-trivial. Since we couldn't design an optimal algorithm, we used the following `Binary-Alloc` heuristic which performed well in practice. The `Binary-Alloc` heuristic is a binary-search-like iterative algorithm as follows. At any iteration, each SU i is assigned a *range* $[l_i, u_i]$ of powers such that if *all* SUs are allocated the lower-end l_i of their ranges then there is no interference caused to any of the PURs. The algorithm iteratively reduces (in half) the range of one of the SUs, till the range of each SU is smaller than the threshold. In particular, in each iteration, the algorithm picks the SU i with the largest range $(u_i - l_i)$, and either moves the lower-bound l_i or upper-bound u_i to $(u_i + l_i)/2$ depending on whether $(u_i + l_i)/2$ causes interference to a PUR.

3) MULTIPLE CHANNELS; REQUEST DURATION

Till now, we have implicitly assumed a single channel. To handle multiple channels, we can extend our schemes easily. If the signal propagation characteristics are different for different channels, then we can create a separate model for each channel. Then, for a single-SU request, we can pick the channel that allows for maximum power allocation. We can also modify the above multi-SU approaches, for the multi-SU scenario, as follows. The `DeepAlloc-Greedy` approach extends naturally. For `DeepAlloc-NN` and `DeepAlloc-RNN`, we use a simple scheme of uniformly (and randomly) partitioning the given set of SUs into multiple sets and assign a channel to each set; incorporating channel assignment in the model would require much more training, and thus not considered for simplicity. Finally, to handle requests for a specified duration, whenever a currently active

SU becomes inactive (i.e., at the end of its authorized duration), we reallocate other active SUs' allocations.

4) OTHER GENERALIZATION: NON-ISOTROPIC USERS, WEATHER, OBSTACLES

We now briefly discuss how our approach can be generalized to handle more general and sophisticated scenarios. We start with discussing how to handle non-isotropic PUs or SUs. In the `PU-Setting`, the non-isotropic PUs can be represented by an appropriate cone (instead of a disk). In the `SS-Setting`, the SS readings would naturally incorporate the impact of non-isotropic PUs—and hence, there are no changes to the approach. To handle changes in propagation models due to different weather conditions, we can train different models for different weather conditions (e.g., rainy, summer, winter, etc). To incorporate more fine-grained weather conditions, we can use a separate sheet to represent weather information over the given area. Note that obstacles and terrain information, being largely fixed, is already implicitly incorporated in the learned model.

IV. LARGE-SCALE SIMULATIONS

In this section, we discuss our large-scale simulation results conducted over a large geographical area (we discuss an outdoor testbed evaluation in the next section). We start with describing the underlying propagation model used in our simulations. All our developed software is open-source [3].

1) LONGLEY-RICE PROPAGATION MODEL AND SETTING

To evaluate our techniques over a realistic propagation model, we use the well-known Longley-Rice [13] Irregular Terrain With Obstruction Model (ITWOM), which is a complex model of wireless propagation based on many parameters including locations, terrain data, obstructions, soil conditions, etc. We use SPLAT! [32] to generate path-loss values; SPLAT! is an open-source software implementing the Longley-Rice propagation model. When simulating the above propagation model, we consider an area of 1km × 1km in our state and use the 600 MHz band to generate path losses using SPLAT!. As the height of an entity is an important factor in determining the path loss, we place the transmitters (PU or SU) at a height of 30m and the receivers (PURs and SSs) at 15m above the ground level.[5] For clarity of presentation and due to limited space, in many plots, we only show results for the `PU-Setting`; in these cases, the observed trend in `SS-Setting` is similar.

2) PERFORMANCE METRICS ($\mathcal{A}_{ERR}$, $\mathcal{A}_{FP}$) AND ALGORITHMS

The main performance metric used to evaluate our technique is the average (absolute) difference in power allocated to the requesting SU with respect to an optimal algorithm;

here, the optimal algorithm has the knowledge of the exact path-loss values and is thus able to use the Eqn. 1 to compute the power to be allocated to the SU. We denote this measure by $\mathcal{A}_{\text{err}}$. To evaluate the impact of false positives, we also consider the average false-positive error metric $\mathcal{A}_{\text{fp}}$ which is computed as the aggregate error in the false-positive samples over the false positives divided by the total number of samples. The $\mathcal{A}_{\text{fp}}$ metric informally measures the level of interference caused by the spectrum allocations. We implement standard ML techniques, viz., a 3-layered neural network (`NN`), support vector regression (`SVR`) [18], our CNN-based approaches `SH-Alloc` (§III-A) and `DeepAlloc` (§III-B). Recall that `DeepAlloc` is based on a pre-trained deep CNN architecture based on ResNet [23] architecture. As detailed in §II, the closest work relevant to our setting is the one in [22] which has available the PUs' parameter values as well as the SS readings, and uses interpolation techniques to estimate path-loss values between relevant entities; we denote this approach by `IP-Based`.[6] We note that the simple `Listen-Before-Talk` approach resulted in a $\mathcal{A}_{\text{err}}$ of a several 10s of dB in our simulations, and has not been shown in the plots for clarity. The large $\mathcal{A}_{\text{err}}$ value for the very conservative `Listen-Before-Talk` approach (see §II-A) is due to the fact that in a large majority of evaluation samples SU is not allocated *any* spectrum power as some of the PUs' power is always received in a large fraction of the area.
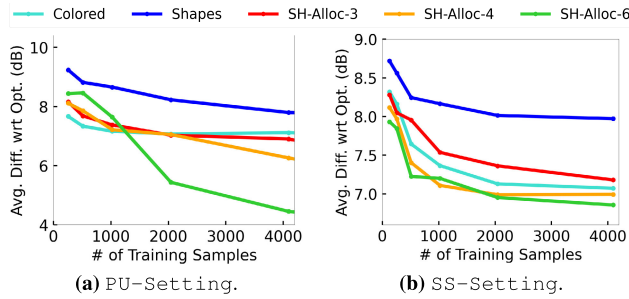
3) TRAINING SAMPLES

As described in §III, there are two types of training samples used in training our models: (i) pre-training samples used to pre-train the `DeepAlloc`; these samples are based on the log-normal propagation model with a path exponent computed from a small number of path-loss samples; we computed an exponent of 3.3 from 200 path-loss samples. (ii) *training samples*, which are gathered in the field as described below; these samples are used to train all the ML models, including the `DeepAlloc` after pre-training. In addition, for our schemes, we also use synthetic samples (§III-E), which are derived from the training samples. To create a training/evaluation sample, we place 10 to 20 PUs at random locations in the given area and assign them a random power within the range of 0 to −30dBm. Each PU has a random number (5 to 10) of PURs distributed randomly within a distance of 50m from the PU. We vary the number of sensors from 49 to 625 (with 400 as the default) sensors uniformly distributed in the field. For `NN` and `SVR` models, we assume the maximum number of PUs to be 20, and use dummy parameter values if the number is less than 20.

Validation and Evaluation Samples. In most of our plots, we vary the total number of training samples from 256 to about 4096 (with a default value of 2048), of which 20%

---

[5]At much lower heights, transient obstacles such as vehicles and temporary structures, would affect the path-loss model—which the SPLAT! software doesn't account for; thus, we choose a higher altitude to simulate an accurate setting.

[6]In our implementation of `IP-Based` algorithm, we fine-tuned its internal path-loss exponent parameter α to get the best performance; we ended up using a value of 3.3.

FIGURE 7. Performance comparison of various schemes for pre-processing the training samples into images.

are used for validation purposes (to tune the model's hyper-parameters) and the remaining are used to train the model. In addition, we create and use 40,000 separate samples for evaluation purposes.

4) TRAINING CNN MODELS

To pre-train the `DeepAlloc` using the generated pre-training samples (we used 1m samples), we used SGD optimizer with a mini-batch size of 64. The learning rate starts from 0.01 and is divided by every 10 epochs. We use a weight decay of 0.00001 and a momentum of 0.9. Then, for fine-tuning the `DeepAlloc` model using real (field) samples, we use *Adam* [28] optimizer with a very low initial learning rate of 0.00001 while keeping the other parameter values the same. Among all the hyper-parameters, we only optimize the regularization value.

A. PERFORMANCE RESULTS (SINGLE SU)

We now present our performance results for various algorithms and settings. We evaluate various techniques for the case of a single SU request; we consider multiple SUs in the next subsection. *Each data point in the plots is an average of over 40,000 evaluation samples.*
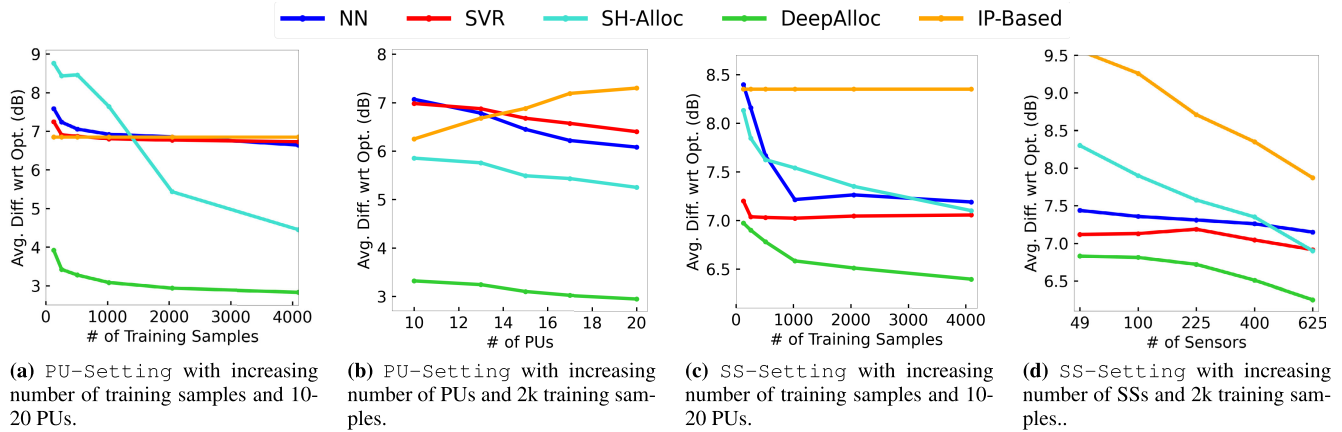
1) VARIOUS PRE-PROCESSING SCHEMES

We start with evaluating various pre-processing schemes for creating images from the training samples, as input to our basic CNN approach `SH-Alloc`. Here, we refer to the multi-sheets based approaches (i.e., from Fig. 2) as `SH-Alloc-N`, where *N* is the number of sheets used for the PUs/SSs. We also evaluate the `Colored` and `Shapes` schemes, where the *Colored* scheme uses different colored-disks (red, green, and blue) for the three entities (PU, SS, and SU) and the `Shapes` scheme uses a grey-scale image with different shapes for the entities (PU: circle, SU: square, SS: rectangle). We maintain the shape sizes to be small (to avoid intersections) and uniform for each entity, but vary the intensity to represent the transmitted (received) power. See Fig. 7 for a performance comparison of the above schemes in both settings. We observe that the `SH-Alloc-N` schemes outperform the other two schemes, with the increase in *N* improving the performance as expected. Thus, we use 6 image sheets for PUs/SSs.
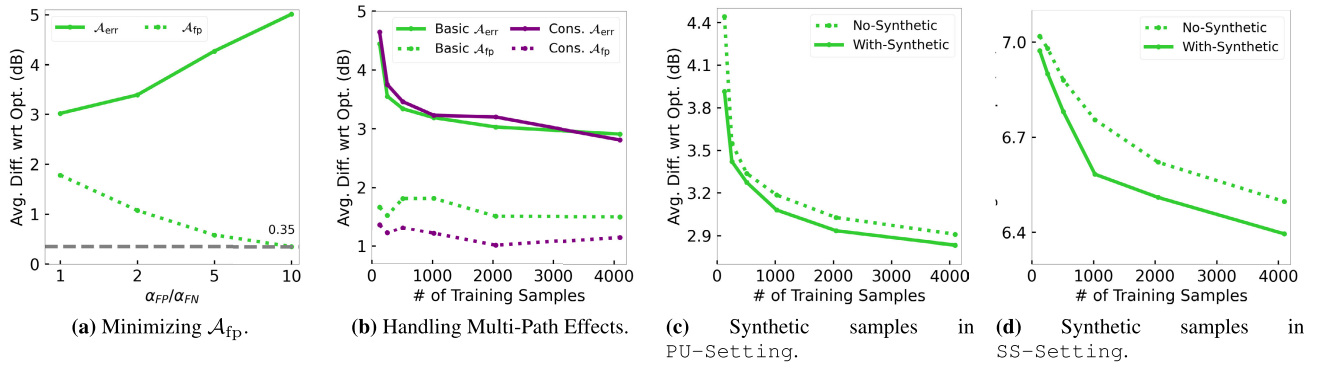
2) VARYING TRAINING COST AND NUMBER OF PUS/SSS

We now evaluate the fundamental performance of various algorithms in terms of the key $\mathcal{A}_{err}$ metric in our two settings, `PU-Setting` and `SS-Setting`, for varying number of (field-gathered) training samples and the number of PUs/SSs. See Fig. 8. First, our main `DeepAlloc` approach outperforms all the other approaches for both settings. More importantly, `DeepAlloc` delivers great performance for the `PU-Setting` with less than 3dB performance gap with the optimal solution. As expected, the performance of our approaches is better in `PU-Setting` than that in `SS-Setting`, since fundamentally the `PU-Setting` has more direct information relevant to the spectrum allocation function. Second and more specifically, we observe that both the CNN-based approaches outperform the `IP-Based` algorithm by a large margin in the `PU-Setting`; in the `SS-Setting`, the `DeepAlloc` approach easily outperforms the `IP-Based` algorithm, except for large number of sensors wherein both techniques perform similarly. Recall that the `IP-Based` approach has the advantage of knowing *both* the PU parameters as well as SS readings, while the ML algorithms have knowledge of only one of these inputs—thus, it is surprising and commendable that `DeepAlloc` is able to outperform the `IP-Based` approach. The above observations suggest that `DeepAlloc` is able to learn the spectrum allocation function effectively. Lastly, we observe that the performance gap between the CNN-based approaches (`SH-Alloc` and `DeepAlloc`) is significant—which demonstrates the value of pre-training a deep CNN model.

3) EVALUATING TECHNIQUES FOR MINIMIZING FALSE-POSITIVE ERRORS, HANDING MULTI-PATH EFFECT, AND SYNTHETIC SAMPLES
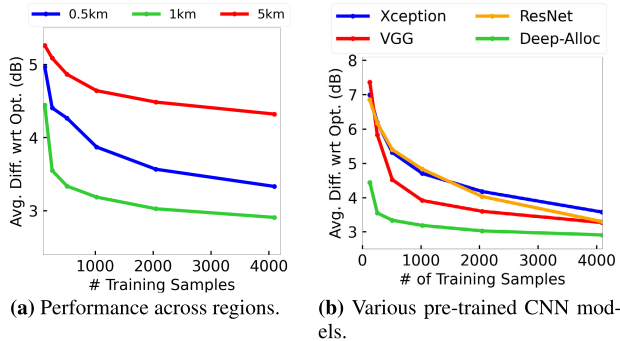
We now evaluate our specialized techniques developed in §III-C-III-E for various aspects. See Fig. 9. Fig. 9(a) evaluates the §III-D's technique to minimize false positives. We plot $\mathcal{A}_{err}$ and $\mathcal{A}_{fp}$ metrics for varying α_{FP}/α_{FN} ratio; we see that increase in α_{FP}/α_{FN} ratio is effective in minimizing the false-positive error $\mathcal{A}_{fp}$, at the cost of higher average $\mathcal{A}_{err}$. Next, in Fig. 9(b), we evaluate the technique from §III-D to handle multi-path fading effects by learning a conservative function via modified labels. Here, we plot the $\mathcal{A}_{err}$ and $\mathcal{A}_{fp}$ for the basic as well as the conservative technique. We observe that the conservative technique's false-positive error ($\mathcal{A}_{fp}$) is indeed reduced without much increase in the total error ($\mathcal{A}_{err}$), compared to the basic approach. Fig. 9(a)-(b) are in `PU-Setting`; the `SS-Setting` results were similar (not shown). Finally, in Fig. 9(c)-(d), we evaluate the benefit of using synthetic samples as described in §III-E. Here, we show `PU-Setting` as well as `SS-Setting` as the algorithms for generating synthetic samples are very different. In `PU-Setting` (Fig. 9(c)), the performance gain due to synthetic samples in positive but minimal—perhaps, because the performance was already near-optimal. In `SS-Setting`

(a) `PU-Setting` with increasing number of training samples and 10-20 PUs.

(b) `PU-Setting` with increasing number of PUs and 2k training samples.

(c) `SS-Setting` with increasing number of training samples and 10-20 PUs.

(d) `SS-Setting` with increasing number of SSs and 2k training samples..

FIGURE 8. Performance comparison of various algorithms for increasing number of training samples and PUs/SSs.

(a) Minimizing $\mathcal{A}_{fp}$.

(b) Handling Multi-Path Effects.

(c) Synthetic samples in `PU-Setting`.

(d) Synthetic samples in `SS-Setting`.

FIGURE 9. (a) Minimizing false-positive error ($\mathcal{A}_{fp}$), (b) Handing multi-path effect via learning a more conservative function, and (c)-(d) Using synthetic samples to improve performance. Here, (a)-(b) are in `PU-Setting`; the `SS-Setting` had similar results.

(a) Performance across regions.

(b) Various pre-trained CNN models.

FIGURE 10. Performance comparison of (a) `DeepAlloc` over different regions, and (b) various pre-trained models for increasing number of training samples.

(Fig. 9(d)), we see a significant performance improvement due to synthetic samples—validating our use of interpolation-based synthetic samples. Here, we added 2500 new uniformly distributed sensors and interpolated their readings using the four nearest original sensors.
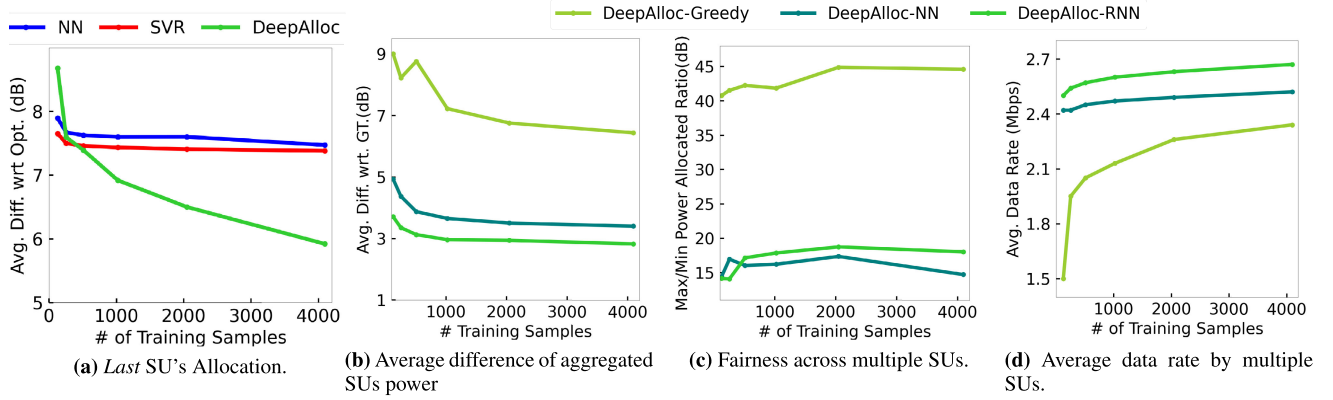
4) MULTIPLE REGIONS

We also evaluated our `DeepAlloc` scheme over other regions with different sizes and terrain characteristics. See Fig. 10(a), which plots performance of `DeepAlloc`

over three different regions—$500m \times 500m$ (our university campus), $1km \times 1km$ (airport landing area, the default region in all other plots), and $5km \times 5km$ (an urban area). We plot results for `PU-Setting` wherein the number of PUs is between 10 and 20. We see that `DeepAlloc` has an $\mathcal{A}_{err}$ of 3-4.5 dB across all regions, and it performs the best over the airport landing area likely due to the lack of buildings and thus more uniform path-loss characteristics.

5) VARIOUS PRE-TRAINED MODELS

Finally, we present evaluation results for various pre-trained deep models. In particular, in addition to our `DeepAlloc` architecture, we also used some well-known Image Classification models such as VGG [44], ResNet [23], and Xception [15] which are all pre-trained with over 1 million images involving our daily-life objects; these pre-trained models were further trained with a varying number of (field-gathered) training samples.[7] See Fig. 10(b), which plots the $\mathcal{A}_{err}$ metric in `PU-Setting` for the above models compared with our `DeepAlloc`. We see that our approach of pre-training using log-normal model-based images in `DeepAlloc` yields a notable performance improvement;

[7]For these models, we use `Colored` scheme of creating images from training images, as they use colored images (rather than sheets) during pre-training.

(a) *Last SU's Allocation.*

(b) *Average difference of aggregated SUs power*

(c) *Fairness across multiple SUs.*

(d) *Average data rate by multiple SUs.*

FIGURE 11. Multiple SUs. (a) Power allocated to subsequent SUs, in presence of other active SUs. (b)-(d) Average different wrt ground truth (GT), Fairness (max/min ratio), and Total data rate for multiple SUs by various multi-SU algorithms.

the performance gap is particularly significant for a lower number of training samples.

6) COMPUTATION TIMES AND COMPLEXITY

The inference time complexity of all our ML approaches is linear in the size of the input, and thus, the inference time in practice is minimal (a fraction of a second). The training time complexity of most ML models depends on the training samples and the resulting convergence, and is thus, uncertain. The actual training times incurred from our set of training samples, on a 4GHz 8-core machine with a GeForce RTX 3080 GPU, were as follows: a few minutes for `NN` as well as `SVR` approaches, and a few hours for `SH-Alloc`. To train the `DeepAlloc` model, it took 5-7 days of computation time to pre-train the model using 1M images and a few hours to train/fine-tune the model after pre-training using SA training samples.
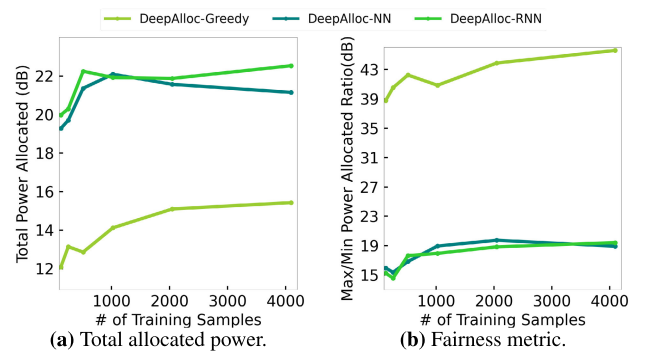
B. MULTIPLE SUS/CHANNELS
1) SUBSEQUENT SU

We start with comparing the spectrum power allocated to a *single* new SU with previous SUs being active. Each sample has a random number (between 1 and 10) of active SUs. For evaluation purposes, we assume that the active SUs are assigned the optimal power (as per Eqn. 1, with full knowledge of PU information and path-loss values) and they transmit using this optimal power. Then, the models predict the allocation power for the last requesting SU. See Fig. 11(a), which shows the performance of various ML algorithms in assigning the power to the last SU. We see that `DeepAlloc` easily outperforms the other ML approaches by a large margin. Also, not shown for clarity, but `IP-Based` had a poor $\mathcal{A}_{err}$ of 21.6dB and `SH-Alloc` had a $\mathcal{A}_{err}$'s of 7.5dB.

2) CONCURRENT SUS: SINGULAR VS. SIMULTANEOUS ALLOCATION

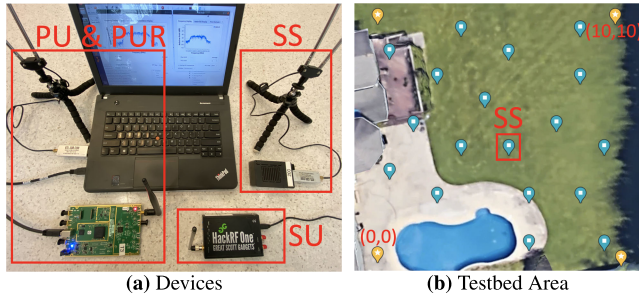We now compare our approaches for handling concurrent SUs (§III-F), viz., `DeepAlloc-Greedy`,

(a) Total allocated power.

(b) Fairness metric.

FIGURE 12. (a) Total allocated power, and (b) Fairness metric, in a four-channel setting with multiple SUs.

`DeepAlloc-NN`, and `DeepAlloc-RNN`; recall that the latter two approaches handle SUs simultaneously. We compare these algorithms in terms of (i) Average absolute-difference wrt to the ground truth (i.e., the output of the `Binary-Alloc` heuristic); (ii) Fairness, in terms of the ratio of maximum to minimum power allocated to an SU; a lower value suggests a more fair allocation. (iii) Overall spectrum utilization, in terms of the total data rate achievable which is estimated as follows. For each SU, we assign a receiver in a random location around the SU (within 100m), calculate the total interference from PUs and other SUs at the receiver, and estimate the achievable data rate using Shannon's capacity law. See Fig. 11(b)-(d). We observe that as expected the `DeepAlloc-NN` and `DeepAlloc-RNN` approaches that allocate powers simultaneously outperform `DeepAlloc-Greedy` significantly in all metrics, with `DeepAlloc-RNN` slightly outperforming the `DeepAlloc-NN` approach too (except in the fairness metric) which is not surprising as `DeepAlloc-RNN` has much more input information.
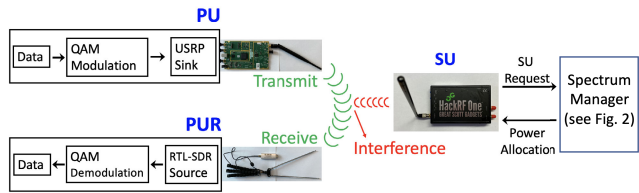
3) MULTIPLE CHANNELS

In Fig. 12, we compare our multi-SU approaches (tailored to multiple channels as described in §III-F) in the 4-channel `PU-Setting` with PUs transmitting over all the channels

(a) Devices

(b) Testbed Area

FIGURE 13. Outdoor testbed. (a) PU, PUR, SU, SS devices. (b) Testbed area (house backyard). Blue stars are the 17 sensors.

FIGURE 14. Testbed system overview, including the GNU-Radio based PU-to-PUR communication system.
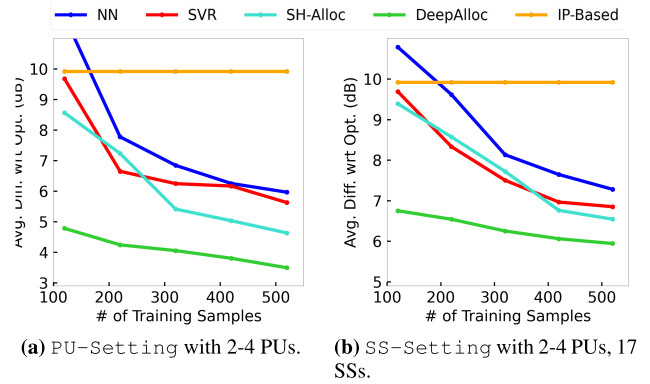
at all times. We allocate powers to 30 SUs. We observe that `DeepAlloc-RNN` still achieves the best performance in terms of total power allocation as well as fairness, with `DeepAlloc-RNN` performing closely. With respect to the single channel, the total power allocated is almost four times (in Watts).

V. `DEEPALLOC` TESTBED IMPLEMENTATION

In this section, we implement and evaluate a complete testbed system for our spectrum allocation system. We use the testbed to collect training samples, which are then used to train and evaluate the learned models. The testbed implementation demonstrates the effectiveness of our techniques in a realistic small-scale setting.

A. TRANSMITTER AND RECEIVER DEVICES USED

We use 4 USRP B200/B210 and 1 HackRF to play the role of the 4 PUs and 1 SU, respectively. A spectrum sensor (SS) is composed of an RTL-SDR dongle that connects to a dipole antenna and is powered by a single-board computer Odroid-C2. We deploy 17 of these spectrum sensors in the testbed. See Fig. 13(a). To simulate PU receivers (i.e., PURs), we use the same RTL-SDR dongle and antenna but power it by a laptop. Each PU is paired with one PUR, and they are both powered by a single laptop. Overall, we implemented a Python repository running on Linux that transmits and receives signals and measures and collects relevant parameters in real-time at 915 MHz ISM band at a sample rate of 1 MHz. We built our custom communication system based on GNU Radio for data communication between PU and PURs, used to determine labels (see below). See Fig. 14.

(a) `PU-Setting` with 2-4 PUs.

(b) `SS-Setting` with 2-4 PUs, 17 SSs.

FIGURE 15. Testbed performance for increasing training set.

B. COLLECTING TRAINING SAMPLES

Recall that a sample in `PU-Setting` is comprised of a sample of PUs' parameters (location and power) and the optimal power allocated to the SU. In `SS-Setting`, a training sample is comprised of spectrum sensors' received power readings. The location of entities is available by using a GPS dongle connected to the laptops as described below, and the sensor's received power is computed as follows. First, we compute an FFT on the I/Q samples collected within a time window to get a power spectral density (PSD) plot. Then, we compute the area under the PSD curve over the 1 MHz channel of interest (see below), and finally, convert the computed area to an appropriate unit.

Determining Labels (Optimal Power Allocated to SU). We essentially do a binary search to estimate the optimal power that can be allocated to SU. To determine whether PU to PUR transmission is incurring any harmful interference from SU, we have PU continuously streaming ASCII messages over the 1 MHz bandwidth channel centered at frequency 915.8 MHz, and check if the messages are successfully received at the PUR. This end-to-end communication system is implemented using GNU Radio.

C. TESTBED AREA AND SETTING

Due to the ongoing pandemic, the testbed was conducted in the backyard of a private house. The whole area is $24m \times 24m$ large, which is similar in size to the testbeds considered in recent works [27], [52] for shared spectrum systems. We divide the area into 100 grid cells where each represents $2.4m \times 2.4m$. See Figure 13(b). We use a GPS dongle that returns the location in (latitude and longitude) and the program converts it into coordinates. We determine the location of PU, SU, and the sensors with the help of GPS dongles and manual observation. All the Odroids and laptops are connected to an outdoor WiFi router and communicate through ssh protocol. For training and evaluation, the 17 sensing devices are placed on the ground and are uniformly spread out. PUs and SU are randomly placed in the area such that different regions of the backyard are "covered". PUs' power is randomly assigned within a certain range.
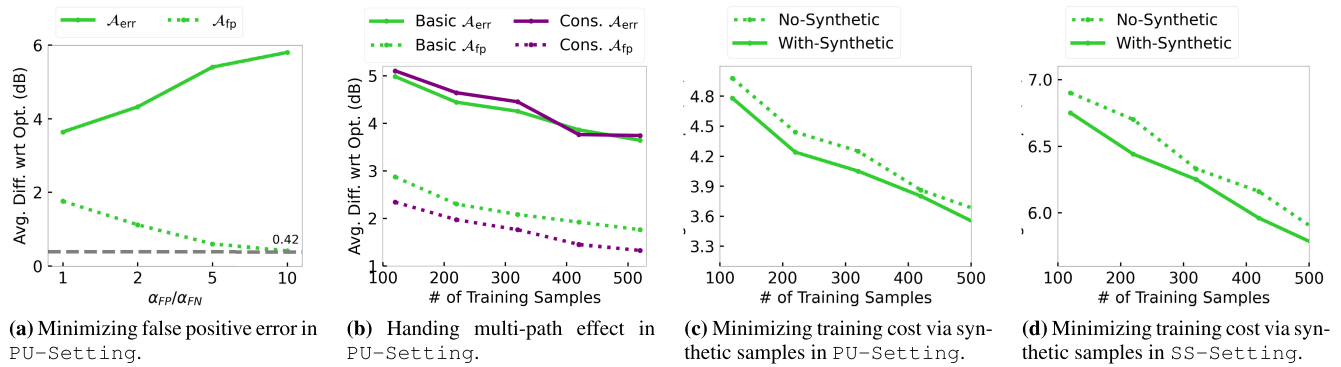
FIGURE 16. Evaluation of various aspects in the testbed.

(a) Minimizing false positive error in `PU-Setting`.

(b) Handing multi-path effect in `PU-Setting`.

(c) Minimizing training cost via synthetic samples in `PU-Setting`.

(d) Minimizing training cost via synthetic samples in `SS-Setting`.

D. RESULTS

First, we evaluate the performance of an increasing number of training samples. See Fig. 15. We observe a similar trend as in the previous section of large-scale simulation, with `DeepAlloc` outperforming other algorithms with a notable margin. In particular, the overall performance of `DeepAlloc` is good, with 4-5dB error using only 500 training samples. The performance of all algorithms is better in `PU-Setting` relative to the `SS-Setting`, as in §IV. Note that `IP-Based` performs quite poorly in this realistic setting, in spite of having knowledge of PU information as well as SS readings, as it assumes an imprecise propagation model. Finally, we evaluate our techniques to handle various aspects, viz., false positive error, multi-path effect, and synthetic samples in Fig. 16. Overall, we observe a similar trend as in the large-scale simulations.

VI. CONCLUSION

We have developed an effective deep-learning technique based on CNNs to learn the spectrum allocation function, and have demonstrated its effectiveness via extensive large-scale simulations as well as a small outdoor testbed. There are many avenues for further improvement of our techniques. First, one could easily pre-train the `DeepAlloc` model with a much larger number of pre-training samples; as our pre-training samples can be automatically generated, this only incurs additional computational cost but no additional field training/deployment cost. Second, we could also use more sophisticated models to generate high-fidelity pre-training samples. Lastly, we could incorporate terrain information in the image sheets to aid the learning process. These directions form the focus of our future work.

REFERENCES

[1] *FCC 3.5GHz Band*. [Online]. Available: https://www.fcc.gov/wireless/bureau-divisions/mobility-division/35-ghz-band/35-ghz-band-overview
[2] *FCC White Space*. [Online]. Available: https://www.fcc.gov/general/white-space
[3] (2023). *DeepAlloc's Source Code Repository*. [Online]. Available: https://github.com/Wings-Lab/SpectrumAllocationDL
[4] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
[5] F. Azmat, Y. Chen, and N. Stocks, "Analysis of spectrum occupancy using machine learning algorithms," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 6853–6860, Sep. 2016.
[6] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2017.
[7] S. Bhattarai, J. J. Park, B. Gao, K. Bian, and W. Lehr, "An overview of dynamic spectrum sharing: Ongoing initiatives, challenges, and a roadmap for future research," *IEEE Trans. Cognit. Commun. Netw.*, vol. 2, no. 2, pp. 110–128, Jun. 2016.
[8] M. M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans, "DIMSUMnet: New directions in wireless networking using coordinated dynamic spectrum," in *Proc. IEEE WOWMOM*, Jun. 2005, pp. 78–85.
[9] S.-S. Byun, I. Balasingham, and X. Liang, "Dynamic spectrum allocation in wireless cognitive sensor networks: Improving fairness and energy efficiency," in *Proc. IEEE 68th Veh. Technol. Conf.*, Sep. 2008, pp. 1–5.
[10] R. Calvo-Palomino, D. Giustiniano, V. Lenders, and A. Fakhreddine, "Crowdsourcing spectrum data decoding," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[11] A. Chakraborty and S. R. Das, "Measurement-augmented spectrum databases for white space spectrum," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2014, pp. 67–74.
[12] A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das, "SpecSense: Crowdsensing for efficient querying of spectrum occupancy," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[13] K. Chamberlin and R. Luebbers, "An evaluation of longley-rice and GTD propagation models," *IEEE Trans. Antennas Propag.*, vol. AP-30, no. 6, pp. 1093–1098, Nov. 1982.
[14] W. Chen and J. Zheng, "A reinforcement learning based joint spectrum allocation and power control algorithm for D2D communication underlaying cellular networks," in *Proc. AICON*, 2019, pp. 146–158.
[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807.
[16] M. Curran, X. Liang, H. Gupta, O. Pandey, and S. Das, "ProCSA: Protecting privacy in crowdsourced spectrum allocation," in *Proc. ESORICS*, 2019, pp. 556–576.
[17] Y. Dou, K. Zeng, H. Li, Y. Yang, B. Gao, K. Ren, and S. Li, "P²-SAS: Privacy-preserving centralized dynamic spectrum access system," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 1, pp. 173–187, 2016.
[18] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. NIPS*, 1997, pp. 1–7.
[19] M. El Tanab and W. Hamouda, "Resource allocation for underlay cognitive radio networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1249–1276, 2nd Quart., 2017.
[20] S. Gao, L. Qian, and D. R. Vaman, "Distributed energy efficient spectrum access in wireless cognitive radio sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2008, pp. 1442–1447.
[21] E. Ghadimi, F. Davide Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.
[22] H. Gupta, M. S. Rahman, and M. Curran, "Shared spectrum allocation via pathloss estimation in crowdsensed shared spectrum systems," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Nov. 2019, pp. 1–5.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[24] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 745–753.

[25] A. T. Hoang, Y.-C. Liang, and M. H. Islam, "Power control and channel allocation in cognitive radio networks with primary users' cooperation," *IEEE Trans. Mobile Comput.*, vol. 9, no. 3, pp. 348–360, Mar. 2010.

[26] P. Karimi, W. Lehr, I. Seskar, and D. Raychaudhuri, "SMAP: A scalable and distributed architecture for dynamic spectrum management," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Oct. 2018, pp. 1–10.

[27] M. Khaledi, M. Khaledi, S. Sarkar, S. Kasera, N. Patwari, K. Derr, and S. Ramirez, "Simultaneous power-based localization of transmitters for crowdsourced spectrum monitoring," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2017, pp. 235–247.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[29] W.-Y. Lee and I. F. Akyldiz, "A spectrum decision framework for cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 2, pp. 161–174, Feb. 2011.

[30] A. E. Leu, M. McHenry, and B. L. Mark, "Modeling and analysis of interference in listen-before-talk spectrum access schemes," *Int. J. Netw. Manag.*, vol. 16, no. 2, pp. 131–147, Mar. 2006.

[31] X. Li and S. A. Zekavat, "Distributed channel assignment in cognitive radio networks," in *Proc. Int. Conf. Wireless Commun. Mobile Comput., Connecting World Wirelessly*, Jun. 2009, pp. 989–993.

[32] J. A. Magliacane. (2008). *SPLAT! A Terrestrial RF Path Analysis Application for Linux/Unix.* [Online]. Available: https://www.qsl.net/kd2bd/splat.html

[33] A. Medeisis and A. Kajackas, "On the use of the universal Okumura–Hata propagation prediction model in rural areas," in *Proc. VTC-Spring. IEEE 51st Veh. Technol. Conf.*, May 2000, pp. 1815–1818.

[34] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.

[35] N. Morozs, "Accelerating reinforcement learning for dynamic spectrum access in cognitive wireless networks," Ph.D. thesis, Dept. Electron., Univ. York, U.K., 2015.

[36] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.

[37] Y. S. Nasir and D. Guo, "Deep reinforcement learning for joint spectrum and power allocation in cellular networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

[38] K. Qiu, S. Bakirtzis, H. Song, I. Wassell, and J. Zhang, "Deep learning-based path loss prediction for outdoor wireless communication systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–2.

[39] V. V. Ratnam, H. Chen, S. Pawar, B. Zhang, C. J. Zhang, Y.-J. Kim, S. Lee, M. Cho, and S.-R. Yoon, "FadeNet: Deep learning-based mm-wave large-scale channel fading prediction and its applications," *IEEE Access*, vol. 9, pp. 3278–3290, 2021.

[40] H. A. Bany Salameh, "Throughput-oriented channel assignment for opportunistic spectrum access networks," *Math. Comput. Model.*, vol. 53, nos. 11–12, pp. 2108–2118, Jun. 2011.

[41] A. Seretis and C. D. Sarris, "An overview of machine learning techniques for radiowave propagation modeling," *IEEE Trans. Antennas Propag.*, vol. 70, no. 6, pp. 3970–3985, Jun. 2022.

[42] P. M. Shankar, *Fading and Shadowing in Wireless Systems*. Berlin, Germany: Springer, 2017.

[43] J. Shi, Z. Guan, C. Qiao, T. Melodia, and G. Challen, "Crowdsourcing access network spectrum allocation using smartphones," in *Proc. HotNets*, 2014, pp. 1–7.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.

[45] Y. Song, K. W. Sung, and Y. Han, "Coexistence of Wi-Fi and cellular with listen-before-talk in unlicensed spectrum," *IEEE Commun. Lett.*, vol. 20, no. 1, pp. 161–164, Jan. 2016.

[46] E. Z. Tragos, S. Zeadally, A. G. Fragkiadakis, and V. A. Siris, "Spectrum assignment in cognitive radio networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1108–1135, 3rd Quart., 2013.

[47] M. Uccellari, F. Facchini, M. Sola, E. Sirignano, G. M. Vitetta, A. Barbieri, and S. Tondelli, "On the application of support vector machines to the prediction of propagation losses at 169 MHz for smart metering applications," *IET Microw., Antennas Propag.*, vol. 12, no. 3, pp. 302–312, Feb. 2018.

[48] H. Wang, J. Ren, and T. Li, "Resource allocation with load balancing for cognitive radio networks," in *Proc. GlobCom*, 2010, pp. 1–5.

[49] W. Wang, K. G. Shin, and W. Wang, "Joint spectrum allocation and power control for multihop cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 7, pp. 1042–1055, Jul. 2011.

[50] Y. Wang, Z. Ye, P. Wan, and J. Zhao, "A survey of dynamic spectrum allocation based on reinforcement learning algorithms in cognitive radio networks," *Artif. Intell. Rev.*, vol. 51, no. 3, pp. 493–506, Mar. 2019.

[51] L. Yu, C. Liu, and W. Hu, "Spectrum allocation algorithm in cognitive ad-hoc networks with high energy efficiency," in *Proc. Int. Conf. Green Circuits Syst.*, Jun. 2010, pp. 349–354.

[52] C. Zhan, H. Gupta, A. Bhattacharya, and M. Ghaderibaneh, "Efficient localization of multiple intruders in shared spectrum system," in *Proc. 19th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2020.

[53] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power control based on deep reinforcement learning for spectrum sharing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4209–4219, Jun. 2020.

[54] X. Zhang, X. Shu, B. Zhang, J. Ren, L. Zhou, and X. Chen, "Cellular network radio propagation modeling with deep convolutional neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2378–2386.

[55] Y. Zhang, C. Kang, T. Ma, Y. Teng, and D. Guo, "Power allocation in multi-cell networks using deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–6.

[56] Q. Zhao and B. M. Sadler, "A survey of dynamic spectrum access," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 79–89, 2007.

MOHAMMAD GHADERIBANEH (Member, IEEE) received the B.S. degree in electrical engineering from Azad University, Urmia, Iran, in 2008, the M.S. degree in telecommunication engineering from Shahid Beheshti University, Tehran, Iran, in 2011, and the Ph.D. degree in computer science from Stony Brook University, NY, USA, in 2023.

From 2012 to 2018, he was in the industry as a Software and Telecommunication Engineer developing wireless communication networks like TETRA. He is currently with Google YouTube. His research interests include quantum networks and machine learning application in wireless networks.

CAITAO ZHAN (Member, IEEE) received the B.S. degree in computer science and technology from the China University of Geosciences, Wuhan, China, in 2017. He is currently pursuing the Ph.D. degree. Then, he joined the Ph.D. Program with the Department of Computer Science, Stony Brook University. His research interests include computer networks, the intersection of wireless networks and machine learning, quantum communication, and quantum sensing.

HIMANSHU GUPTA (Member, IEEE) received the B.Tech. degree in computer science and engineering from IIT Bombay, in 1992, and the M.S. and Ph.D. degrees in computer science from Stanford University, in 1999. He is currently a Professor in computer science with Stony Brook University, where he has been a Faculty Member, since 2002. His research interests include wireless networks, with a recent focus on free-space optical communication networks and spectrum management. His current research interests include quantum networks and communication and distributed quantum algorithms.

• • •