Data-Driven Spectrum Cartography via Deep Completion Autoencoders

Yves Teganya and Daniel Romero

Abstract **—Spectrum maps, which provide RF spectrum metrics such as power spectral density for every location in a geographic area, find numerous applications in wireless communications such as interference control, spectrum management, resource allocation, and network planning to name a few. Spectrum cartography techniques construct these maps from a collection of measurements collected by spatially distributed sensors. Due to the nature of the propagation of electromagnetic waves, spectrum maps are complicated functions of the spatial coordinates. For this reason, model-free approaches have been preferred. However, all existing schemes rely on some interpolation algorithm unable to learn from data. This paper proposes a novel approach to spectrum cartography where propagation phenomena are learned from data. The resulting algorithms can therefore construct a spectrum map from a significantly smaller number of measurements than existing schemes since the spatial structure of shadowing and other phenomena is previously learned from maps in other environments. Besides the aforementioned new paradigm, this is also the first work to perform spectrum cartography with deep neural networks. To exploit the manifold structure of spectrum maps, a deep network architecture is proposed based on completion autoencoders.**

Keywords— Spectrum cartography, deep learning, cognitive radio, completion autoencoders.

C.1 Introduction

Spectrum cartography constructs maps of RF channel metrics such as received signal power, interference power, power spectral density (PSD), electromagnetic absorption, or channel gain; see e.g. [1–3]. Besides applications like source localization [2] or radio tomography [4, 5], spectral maps find a myriad of applications in wireless communications such as network planning, interference coordination, power control, spectrum management, resource allocation, handoff procedure design, dynamic spectrum access, and cognitive ratio [6–8]. Spectrum maps are constructed from measurements acquired by spectrum sensors or mobile devices.

Most approaches are based on some interpolation algorithm. For example, power maps have been constructed through kriging [1,9], dictionary learning [10,11], compressive sensing [3], Bayesian models [12], matrix completion [13], and kernel methods [14, 15]. PSD maps have also been constructed by exploiting the sparsity of power across space and frequency [2] as well as by applying thin-plate spline regression [16] and kernel-based learning [8, 17]. Metrics other than power and PSD have also been mapped in the literature. For example, [5, 18, 19] are capable of constructing channel gain maps.

Unfortunately, none of the existing approaches can learn from data. This means that they fail to learn the characteristics of the propagation phenomena and, therefore, a substantial performance improvement is expected if such knowledge can be incorporated.

To address this limitation, the first contribution of this paper is a data-driven paradigm for spectrum cartography. Specifically, it proposes learning the spatial features of the relevant propagation phenomena such as shadowing, reflection, and diffraction using a data set of past measurements. Intuitively, leveraging these learned features can significantly reduce the number of measurements required to attain a target performance. This aspect is critical since all measurements need to be collected in a sufficiently short time since the mapped metric is subject to temporal variations in real-world scenarios. The second contribution comprises a spectrum cartography algorithm to construct PSD maps relying on a deep neural network. Although several approaches for applying this class of networks are discussed, the most natural one relies on a spatial discretization of the area of interest. The resulting tensor completion task is addressed by means of a *completion network* architecture with an encoder-decoder structure that capitalizes on the observation that spectrum maps lie close to a low-dimensional manifold embedded in a high-dimensional space. Our experiments reveal that the performance of such algorithm beats the state-of-the-art alternatives. Finally, all code, trained networks, and the data set constructed for this work will be posted at the authors' web sites.

The novelty of this paper is twofold. First, this is the first work to propose a data-driven spectrum cartography approach. Second, this is the first work to propose a deep learning algorithm for spectrum cartography.

The rest of this paper is organized as follows. Sec. C.2 describes the problem of PSD cartography. Sec. C.3 presents the aforementioned data-driven spectrum cartography paradigm and proposes a deep neural network architecture based on completion autoencoders. Simulations and conclusions are respectively provided in Secs. C.4 and C.5. Unfortunately, due to space requirements, we had to omit many insightful explanations and a large number of experiments that further support the proposed approach.

Notation: $|\mathcal{A}|$ denotes the cardinality of set $\mathcal{A}$. $[\boldsymbol{A}]_{i,j}$ is the (i,j)-th entry of matrix $\boldsymbol{A}$, whereas $[\boldsymbol{B}]_{i,j,k}$ is the (i,j,k)-th entry of tensor $\boldsymbol{B}$. Finally, $\boldsymbol{A}^\top$ is the transpose of matrix $\boldsymbol{A}$.

C.2 Model and Problem Formulation

Consider L sources located in a geographical region of interest $\mathcal{X} \subset \mathbb{R}^2$ and operating on a certain frequency band. Let $\Upsilon_l(f)$ denote the transmit PSD of the l-th source and let $H_l(\boldsymbol{x}, f)$ represent the frequency response of the channel between the l-th source and a sensor with an isotropic antenna located at $\boldsymbol{x} \in \mathcal{X}$. For simplicity, assume that small-scale fading has been averaged out; see also Remark 5. Both $\Upsilon_l(f)$ and $H_l(\boldsymbol{x}, f)$ are assumed to remain constant over time, a realistic assumption provided that the measurements described below are collected in an interval of shorter length than the channel coherence time and time scale of changes in $\Upsilon_l(f)$.

If the L signals are uncorrelated, the PSD at $\boldsymbol{x} \in \mathcal{X}$ is

$$\Psi(\boldsymbol{x}, f) = \sum_{l=1}^{L} \Upsilon_l(f)|H_l(\boldsymbol{x}, f)|^2 + \upsilon(\boldsymbol{x}, f)$$

with $\upsilon(\boldsymbol{x}, f)$ the noise PSD of a generic sensor at location $\boldsymbol{x}$, which models thermal noise, background radiation noise, and interference from remote sources. A certain number of devices, such as mobile users in a cellular communication network or spectrum sensors, collect PSD measurements $\{\tilde{\Psi}(\boldsymbol{x}_n, f)\}_{n=1}^{N}$ at N locations $\{\boldsymbol{x}_n\}_{n=1}^{N} \subset \mathcal{X}$ and finite set of frequencies $f \in \mathcal{F}$; see also Remark 6. These measurements can be obtained using e.g. periodograms or spectral analysis methods such as the Bartlett or Welch method [20].

These measurements are sent to a fusion center, which may be e.g. a base station, a mobile user, or a cloud server, depending on the application. Given $\{(\boldsymbol{x}_n, \tilde{\Psi}(\boldsymbol{x}_n, f)),\ n = 1, \ldots, N,\ f \in \mathcal{F}\}$, the fusion center must obtain an estimate $\hat{\Psi}(\boldsymbol{x}, f)$ of $\Psi(\boldsymbol{x}, f)$ at every location $\boldsymbol{x} \in \mathcal{X}$ and frequency $f \in \mathcal{F}$. In spectrum cartography, function $\Psi(\boldsymbol{x}, f)$ is typically referred to as the *true map*, whereas $\hat{\Psi}(\boldsymbol{x}, f)$ is the *estimated map* or *map estimate*. The algorithm or rule that provides a map estimate, which in this paper is a neural network, is termed *map estimator*. The challenge is to exploit the spatial structure of propagation phenomena so that the estimation error, quantified e.g. as $\sum_f \int_{\mathcal{X}} |\Psi(\boldsymbol{x}, f) - \hat{\Psi}(\boldsymbol{x}, f)|^2 d\boldsymbol{x}$, is minimized for a certain N or, alternatively, the minimum N required to guarantee a target estimation error is minimized.

To the best of our knowledge, all existing approaches to spectrum cartography are based on interpolation algorithms that do not learn from data. In contrast, the next section develops a novel data-driven methodology that learns the aforementioned structure from a data set.

Remark 3 *Sensors must determine their locations $\{\boldsymbol{x}_n\}_n$ with an error sufficiently small relative to the scale of spatial variability of $\Psi(\boldsymbol{x}, f)$ across $\mathcal{X}$. Thus, estimating small-scale fading is more challenging than estimating shadowing since the coherence distance of the former is comparable to the wavelength and typical communication bands of interest have wavelengths in the order of centimeters.*

Remark 4 *The number of measurement locations may be significantly larger than the number of sensors if the sensors move. Measurements collected at different locations may be useful to estimate a spectrum map so long as the difference between measurement instants is small relative to the time scale of the variations of the PSD map. The latter*

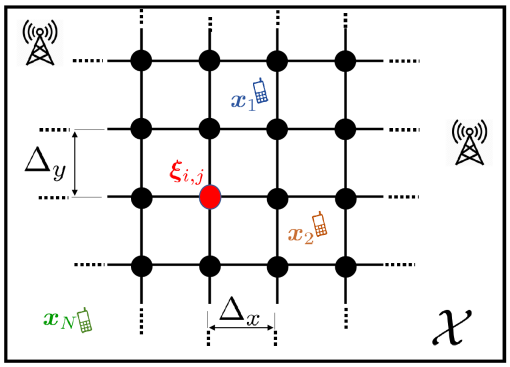

Figure C.1: Model setup and area discretization.

is highly dependent on the specific application. For example, one expects significant variations in DVB-T bands to occur in the scale of several months, whereas PSD maps in LTE bands may change in the scale of milliseconds due to power control, mobility, and interference.

C.3 Proposed Data-Driven Cartography

This section introduces a data-driven paradigm for spectrum cartography and develops a deep learning algorithm that abides by this principle. To this end, Sec. C.3.1 starts by reformulating the problem at hand as a tensor-completion task amenable to application of deep neural networks. Subsequently, Sec. C.3.2 addresses unique aspects of tensor/-matrix completion via deep learning. Finally, Secs. C.3.3 and C.3.4 respectively describe how a deep neural network can be trained to learn the spatial structure of propagation phenomena and how this task can be addressed via the notion of *completion autoencoders*.

C.3.1 Spectrum Cartography as a Tensor Completion Task

Observe that the value of N depends on the number and movement of the sensors relative to the time-scale of temporal changes in $\Psi(\boldsymbol{x}, f)$; cf. Sec. C.2. In principle, one could think of using a separate map estimator for each possible value of N. Each estimator could be relatively simple since it would always take the same number of inputs. However, such an approach would be highly inefficient in terms of memory, computation, and prone to erratic behavior since each estimator is trained with a different data set. Thus, it is more practical to rely on a single estimator that can accommodate arbitrary values of N. A customary approach in deep learning for coping with inputs of variable lengths is through recurrent neural networks [21], [22, Ch. 10]. Unfortunately, besides the difficulties of training these networks, it is unclear how such an approach could effectively exploit spatial information. For this reason, the selected approach in this work is to reformulate the cartography problem as a tensor completion task amenable to a solution based on a feedforward architecture [22, Ch. 6].

To this end, one must discretize $\mathcal{X}$, a trick already applied in radio tomographic imaging [23, 24] and spectrum cartography [13]. To introduce the appropriate notation,

it will be briefly outlined next. Define an $N_y \times N_x$ rectangular grid over $\mathcal{X}$, as depicted in Fig. C.1. This grid comprises points $\boldsymbol{\xi}_{i,j}$ evenly spaced by Δ_x and Δ_y along the x- and y-axes respectively, that is, the (i,j)-th grid point is given by $\boldsymbol{\xi}_{i,j} := [i\Delta_x,\ j\Delta_y]^\top$, with $i = 1,\ldots,N_y,\ j = 1,\ldots,N_x$. For future usage, define $\mathcal{A}_{i,j} \subset \{1,\ldots,N\}$ as the set containing the indices of the measurement locations assigned to the (i,j)-th grid point by the criterion of minimum distance, i.e., $n \in \mathcal{A}_{i,j}$ iff $||\boldsymbol{\xi}_{i,j} - \boldsymbol{x}_n|| \leq ||\boldsymbol{\xi}_{i',j'} - \boldsymbol{x}_n|| \forall i', j'$ with $i' \neq i$ and $j' \neq j$.

This grid induces a discretization of $\Psi(\boldsymbol{x}, f)$ along the $\boldsymbol{x}$ variable. One can therefore collect the true PSD values at the grid points in matrix $\boldsymbol{\Psi}(f) \in \mathbb{R}^{N_y \times N_x}$, $f \in \mathcal{F}$, whose (i,j)-th entry is given by $[\boldsymbol{\Psi}(f)]_{i,j} = \Psi(\boldsymbol{\xi}_{i,j}, f)$. By letting $\mathcal{F} = \{f_1,\ldots,f_{N_f}\}$, it is also possible to stack these matrices along the third dimension to form the tensor $\boldsymbol{\Psi} \in \mathbb{R}^{N_y \times N_x \times N_f}$, where $[\boldsymbol{\Psi}]_{i,j,n_f} = \Psi(\boldsymbol{\xi}_{i,j}, f_{n_f})$, $n_f = 1,\ldots,N_f$. For short, the term *true map* will either refer to $\Psi(\boldsymbol{x}, f)$ or $\boldsymbol{\Psi}$.

Similarly, one can collect the measurements in a tensor of the same dimensions. Informally, if the grid is sufficiently fine (Δ_x and Δ_y are sufficiently small), it holds that $\boldsymbol{x}_n \approx \boldsymbol{\xi}_{i,j}\ \forall n \in \mathcal{A}_{i,j}$ and, correspondingly, $\Psi(\boldsymbol{x}_n, f) \approx \Psi(\boldsymbol{\xi}_{i,j}, f)\ \forall n \in \mathcal{A}_{i,j}$. It follows that,

$$\Psi(\boldsymbol{\xi}_{i,j}, f) \approx \frac{1}{|\mathcal{A}_{i,j}|} \sum_{n \in \mathcal{A}_{i,j}} \Psi(\boldsymbol{x}_n, f)$$

whenever $|\mathcal{A}_{i,j}| \geq 1$. Therefore, it makes sense to aggregate all the measurements assigned to $\boldsymbol{\xi}_{i,j}$ as[1]

$$\tilde{\Psi}(\boldsymbol{\xi}_{i,j}, f) := \frac{1}{|\mathcal{A}_{i,j}|} \sum_{n \in \mathcal{A}_{i,j}} \tilde{\Psi}(\boldsymbol{x}_n, f).$$

Conversely, when $|\mathcal{A}_{i,j}| = 0$, there are no measurements associated with $\boldsymbol{\xi}_{i,j}$, in which case one says that there is a *miss* at $\boldsymbol{\xi}_{i,j}$. Upon letting $\Omega \subset \{1,\ldots,N_y\} \times \{1,\ldots,N_x\}$ be such that $(i,j) \in \Omega$ iff $|\mathcal{A}_{i,j}| > 0$, all (possibly aggregated) measurements $\tilde{\Psi}(\boldsymbol{\xi}_{i,j}, f)$ can be collected in $\tilde{\boldsymbol{\Psi}}(f) \in \mathbb{R}^{N_y \times N_x}$, defined as

$$[\tilde{\boldsymbol{\Psi}}(f)]_{i,j} = \begin{cases} \tilde{\Psi}(\boldsymbol{\xi}_{i,j}, f) & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

Note that misses have been filled with zeroes, but other values can be used.

When $(i,j) \in \Omega$, the values of $[\tilde{\boldsymbol{\Psi}}(f)]_{i,j}$ and $[\boldsymbol{\Psi}(f)]_{i,j}$ differ due to the error introduced by the spatial discretization as well as due to the measurement error incurred when measuring $\Psi(\boldsymbol{x}_n, f),\ n \in \mathcal{A}_{i,j}$. The latter is caused mainly by the finite time devoted by sensors to take measurements.

As before, the matrices $\tilde{\boldsymbol{\Psi}}(f),\ f = 1,\ldots,N_f$ can be stacked along the 3rd dimension to form $\tilde{\boldsymbol{\Psi}} \in \mathbb{R}^{N_y \times N_x \times N_f}$, where $[\tilde{\boldsymbol{\Psi}}]_{i,j,n_f} = [\tilde{\boldsymbol{\Psi}}(f_{n_f})]_{i,j}$. For short, this tensor will be referred to as the *sampled map*.

The cartography problem stated in Sec. C.2 can now be reformulated as, given Ω and $\tilde{\boldsymbol{\Psi}}$, estimate $\boldsymbol{\Psi}$.

[1]For simplicity, the notation implicitly assumes that $\boldsymbol{x}_n \neq \boldsymbol{\xi}_{i,j}\ \forall n, i, j$, but this is not a requirement.

C.3.2 Feedforward Completion Networks

The previous section reformulated the spectrum cartography problem as a tensor completion task. Since conventional neural networks cannot directly accommodate input misses and set-valued inputs like Ω, this section explores the possibilities and motivates the adopted approach.

But before that, a quick refresh on deep learning is in order. A feedforward deep neural network is a function $p_{\boldsymbol{w}}$ that can be expressed as the composition $p_{\boldsymbol{w}}(\boldsymbol{\Phi}) = p^{(L)}_{\boldsymbol{w}_L}(p^{(L-1)}_{\boldsymbol{w}_{L-1}}(\dots p^{(1)}_{\boldsymbol{w}_1}(\boldsymbol{\Phi})))$ of *layer* functions $p^{(l)}_{\boldsymbol{w}_l}$, where $\boldsymbol{\Phi}$ is the input. Although there is no commonly agreed definition of layer function, it is typically formed by concatenating simple scalar-valued functions termed *neurons* that implement a linear function followed by a simple non-linear function known as activation [22]. The term *neuron* stems from the resemblance between these functions and certain simple functional models for natural neurons. Similarly, there is no consensus on which values of L qualify for $p_{\boldsymbol{w}}$ to be considered a *deep* neural network. With vector $\boldsymbol{w}_l$ containing the parameters of the l-th layer, the parameters of the entire network can be collected in $\boldsymbol{w}{:=}[\boldsymbol{w}_1^\top, \dots, \boldsymbol{w}_L^\top]^\top \in \mathbb{R}^{N_w}$. These parameters are *learned* using a *training set* in a process termed *training*.

The rest of this section as well as Sec. C.3.3 carefully delineate how a deep neural network can be trained to perform data-driven spectrum cartography. Occasional references to works in areas such as collaborative filtering and image inpainting will provide insight and motivate the design decisions. On the other hand, Sec. C.3.4 will address the design of $p_{\boldsymbol{w}}$.

Although the training set construction is detailed in Sec. C.3.3, suppose by now that a set of T *training examples* $\{(\tilde{\boldsymbol{\Psi}}_t, \Omega_t)\}_{t=1}^T$ is given. Here, $\{\tilde{\boldsymbol{\Psi}}_t\}_t$ is a collection of sampled maps acquired in different environments and Ω_t the corresponding sampling set.

The desired estimator should obtain $\boldsymbol{\Psi}$ as a function of $\tilde{\boldsymbol{\Psi}}$ and Ω. But regular neural networks cannot directly accommodate set-valued inputs and missing entries. For this reason, [25] proposes filling the missing entries in $\tilde{\boldsymbol{\Psi}}$ by solving

$$\begin{aligned} \underset{\{\boldsymbol{\chi}_t\}_t, \boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{T}\sum_{t=1}^{T} \left\| \mathcal{P}_{\Omega_t}\left(\boldsymbol{\chi}_t - p_{\boldsymbol{w}}(\boldsymbol{\chi}_t)\right)\right\|_F^2, \\ & [\boldsymbol{\chi}_t]_{i,j,n_f} = [\tilde{\boldsymbol{\Psi}}_t]_{i,j,n_f} \ \forall n_f, \forall (i,j) \in \Omega_t, \end{aligned} \tag{C.1}$$

where $||\boldsymbol{A}||_F^2 {:=} \sum_{i,j,n_f} [\boldsymbol{A}]_{i,j,n_f}^2$ is the Frobenius norm of tensor $\boldsymbol{A}$ and $\mathcal{P}_\Omega(\boldsymbol{A})$ is defined as $[\mathcal{P}_\Omega(\boldsymbol{A})]_{i,j,n_f} = [\boldsymbol{A}]_{i,j,n_f}$ if $(i,j) \in \Omega$ and $[\mathcal{P}_\Omega(\boldsymbol{A})]_{i,j,n_f} = 0$ otherwise. The map estimate produced by this method is directly the minimizer $\boldsymbol{\chi}_t$ of (C.1). Observe that if there exists a value of $\boldsymbol{w}$ for which $p_{\boldsymbol{w}}$ becomes the identity map, i.e. $\boldsymbol{\chi} = p_{\boldsymbol{w}}(\boldsymbol{\chi})$, $\forall \boldsymbol{\chi}$, then the optimum of (C.1) is attained regardless of the value of the entries $[\boldsymbol{\chi}_t]_{i,j,n_f}$, $(i,j) \notin \Omega_t$, which would render this estimator useless. Thus, some form of *capacity/complexity control* is necessary [26]. For instance, one can (i) impose constraints on $\boldsymbol{w}$, (ii) add a regularization term to the objective function, or (iii) limit capacity through the design of the network architecture. Approach (iii) will be discussed further in Sec. C.3.4. To simplify the exposition, expressions in this paper will not display constraints or regularizers, but it is understood that the user may include them if necessary.

After $\boldsymbol{w} = \hat{\boldsymbol{w}}$ has been obtained by applying (C.1) with sufficiently large T, one can complete further tensors $\tilde{\boldsymbol{\Psi}}_t$ by setting $\boldsymbol{w}$ in (C.1) to this learned vector $\hat{\boldsymbol{w}}$ and optimize only with respect to $\{\boldsymbol{\chi}_t\}_t$, which is computationally simpler.

The number of optimization variables in (C.1) is $N_w + N_x N_y N_f T$, where N_w is the length of $\boldsymbol{w}$. This number is prohibitive for high T, as required for training deep neural networks. Besides, even with the aforementioned simplified approach that only optimizes with respect to $\{\boldsymbol{\chi}_t\}_t$, a large number of forward and backward backpropagation passes [22, Ch. 6] are required to estimate each map. Thus, this approach is not suitable for real-time implementation, as required in spectrum cartography applications.

To alleviate this limitation, a simple alternative would be to just feed $\tilde{\boldsymbol{\Psi}}$ to the neural network and train by solving

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \tfrac{1}{T}\sum_{t=1}^{T} \left\| \mathcal{P}_{\Omega_t}\left(\tilde{\boldsymbol{\Psi}}_t - p_{\boldsymbol{w}}(\tilde{\boldsymbol{\Psi}}_t)\right)\right\|_F^2. \tag{C.2}$$

Although the missing entries were filled with zeros in Sec. C.3.1, one can alternatively use other real numbers. After (C.2) is solved, $\tilde{\boldsymbol{\Psi}}$ can be completed just by evaluating $p_{\boldsymbol{w}}(\tilde{\boldsymbol{\Psi}})$, which requires a single forward pass. Besides, solving (C.2) involves just N_w optimization variables. However, because the completion step $p_{\boldsymbol{w}}(\tilde{\boldsymbol{\Psi}})$ does not involve Ω, poor performance is expected since the network cannot distinguish missing entries from measurements close to the filling value.

In the application at hand, one could circumvent this limitation by expressing the entries of $\tilde{\boldsymbol{\Psi}}$ in natural power units (e.g. Watt) and filling the misses with a negative number such as -1. Unfortunately, the usage of finite-precision arithmetic would introduce large errors in the map estimates and is problematic in our experience. For this reason, expressing $\tilde{\boldsymbol{\Psi}}$ in logarithmic units such as dBm is preferable. However, the problem of distinguishing missing entries persists since logarithmic units are not confined to be non-negative.

A more practical approach is to complement the input of the network with a binary mask that indicates which entries are observed, as proposed in the image inpainting literature [27]. In this case, the binary mask $\boldsymbol{M}_\Omega \in \{0,1\}^{N_y \times N_x}$ associated with the sampling set Ω is given by $[\boldsymbol{M}_\Omega]_{i,j} = 1$ if $(i,j) \in \Omega$ and $[\boldsymbol{M}_\Omega]_{i,j} = 0$ otherwise.

To simplify notation, let $\check{\boldsymbol{\Psi}} \in \mathbb{R}^{N_y \times N_x \times N_f + 1}$ denote a tensor obtained by concatenating $\tilde{\boldsymbol{\Psi}}$ and $\boldsymbol{M}_\Omega$ along the third dimension. The neural network can therefore be trained as

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \tfrac{1}{T}\sum_{t=1}^{T} \left\| \mathcal{P}_{\Omega_t}\left(\tilde{\boldsymbol{\Psi}}_t - p_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}}_t)\right)\right\|_F^2 \tag{C.3}$$

and, afterwards, a tensor $\tilde{\boldsymbol{\Psi}}$ can be completed just by evaluating $p_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}})$. Then, this scheme is simple to train, inexpensive to test, and exploits information about the location of the misses.

C.3.3 Learning in Real-World Scenarios

A key novelty in this paper is to obtain map estimators by learning from data. This section describes how to construct a suitable training set in the application at hand.

The first consideration pertains to ill-conditioning issues arising when the number of frequencies N_f in $\mathcal{F}$ is large, as will typically be the case. Suppose that the first layer of $p_{\boldsymbol{w}}$ is fully connected and has N_N neurons. Its total number of parameters becomes $(N_y N_x N_f + 1)N_N$ plus possibly additional parameters of the activation functions. Other layers will experience the same issue to different extents. Since T must be comparable to the number of unknowns to train the network effectively, the impact of a large N_f is to drastically limit the number of layers or neurons that can be used.

Previous approaches in spectrum cartography experienced similar issues, which were often addressed by the introduction of parametric models along the frequency domain; see e.g. [8, 16]. Although such an approach can be similarly adopted in the present work, thereby reducing the number of *channels* at the neural network input from $N_f + 1$ to a much smaller number, it will be argued next that directly separating the problem across frequencies may be preferable when training a deep neural network. The idea is that propagation phenomena at similar frequencies are expected to be similar. Building upon this principle, $p_{\boldsymbol{w}}$ can operate separately at each frequency f. This means that training can be accomplished through

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \frac{1}{TN_f} \sum_{t=1}^{T} \sum_{f \in \mathcal{F}} \left\| \mathcal{P}_{\Omega_t} \left(\tilde{\boldsymbol{\Psi}}_t(f) - p_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}}_t(f)) \right) \right\|_F^2, \tag{C.4}$$

where $\check{\boldsymbol{\Psi}}_t(f) \in \mathbb{R}^{N_y \times N_x \times 2}$ is a tensor with first frontal slab given by $\tilde{\boldsymbol{\Psi}}_t(f)$ and second frontal slab given by $\boldsymbol{M}_{\Omega_t}$.

Observe that the number of variables is now reduced by a factor of N_f whereas the "effective" number of training examples has been multiplied by N_f; cf. number of summands in (C.4). This is a drastic improvement especially when N_f takes values such as 512 or 1024, as customary in spectral analysis. Thus, such a frequency separation allows an increase in the number of neurons per layer or (typically more useful [22, Ch. 5]) the total number of layers for a given T. Although such a network would not exploit structure across the frequency domain, the fact that it would be better trained is likely to counteract this limitation in many setups.

The next step is to construct the data set, for which three approaches are discussed next:

C.3.3.1 Synthetic Training Data

Since collecting a large number of training maps may be slow or expensive, one can instead generate maps using a mathematical model or simulator that captures the structure of the propagation phenomena; see e.g. [28]. Fitting $p_{\boldsymbol{w}}$ to data generated by that model could, in principle, yield an estimator that effectively exploits the path loss and shadowing structure. The idea is therefore to generate T maps $\{\Psi_t(\boldsymbol{x}, f)\}_{t=1}^T$ together with T sampling sets $\{\Omega_t\}_{t=1}^T$. Afterwards, $\{\tilde{\boldsymbol{\Psi}}_t\}_{t=1}^T$ and $\{\check{\boldsymbol{\Psi}}_t\}_{t=1}^T$ can be formed as described earlier. It is possible to add artificially generated noise to the synthetic measurements in $\check{\boldsymbol{\Psi}}_t$ to model the effect of measurement error. This would train the network to counteract the impact of such error, along the lines of denoising autoencoders [22, Ch. 14]. The advantage of this approach is that one has access to the ground truth, i.e., one can use

the true maps $\boldsymbol{\Psi}_t$ as *targets*. Specifically, the neural network can be trained on the data $\{(\check{\boldsymbol{\Psi}}_t, \boldsymbol{\Psi}_t)\}_{t=1}^T$ by solving

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \frac{1}{TN_f} \sum_{t=1}^{T} \sum_{f \in \mathcal{F}} \left\| \boldsymbol{\Psi}_t(f) - p_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}}_t(f)) \right\|_F^2 . \tag{C.5}$$

If the model or simulator is sufficiently close to the reality, completing a real-world map $\check{\boldsymbol{\Psi}}(f)$ as $p_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}}(f))$ should produce an accurate estimate.

C.3.3.2 Real Training Data

In practice, real maps may be available for training. However, in most cases, it will not be possible to collect measurements at all grid points before the map changes. Besides, it is not possible to obtain the entries of $\boldsymbol{\Psi}$ but only measurements of it. This means that a real training set is of the form $\{\check{\boldsymbol{\Psi}}_t,\ t = 1, \ldots, T\}$.

For training, one can plug this data directly into (C.4). However, $p_{\boldsymbol{w}}$ may then focus on learning just the values $\{[\tilde{\boldsymbol{\Psi}}_t(f)]_{i,j},\ (i,j) \in \Omega_t\}$, as would happen e.g. when $p_{\boldsymbol{w}}$ is the identity mapping. To counteract this trend, one can use one part of the measurements as the input and another part as the output (target). For each t, construct the Q_t pairs of (not necessarily disjoint) subsets $\Omega_{t,q}^{(I)}, \Omega_{t,q}^{(O)} \subset \Omega_t$, $q = 1, \ldots, Q_t$, e.g by drawing a given number of elements of Ω_t uniformly at random without replacement. Using these subsets, subsample $\tilde{\boldsymbol{\Psi}}_t(f)$ to yield $\tilde{\boldsymbol{\Psi}}_{t,q}^{(I)}(f) {:=} \mathcal{P}_{\Omega_{t,q}^{(I)}}(\tilde{\boldsymbol{\Psi}}_t(f))$ and $\tilde{\boldsymbol{\Psi}}_{t,q}^{(O)}(f) {:=} \mathcal{P}_{\Omega_{t,q}^{(O)}}(\tilde{\boldsymbol{\Psi}}_t(f))$. With these $TN_f \sum_t Q_t$ training examples, one can think of training as

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \tfrac{1}{TN_f \sum_t Q_t} \sum_{f \in \mathcal{F}} \sum_{t=1}^{T} \sum_{q=1}^{Q_t} \left\| \mathcal{P}_{\Omega_{t,q}^{(O)}} \left(\tilde{\boldsymbol{\Psi}}_{t,q}^{(O)}(f) - p_{\boldsymbol{w}} \left(\check{\boldsymbol{\Psi}}_{t,q}^{(I)}(f) \right) \right) \right\|_F^2 , \tag{C.6}$$

where $\check{\boldsymbol{\Psi}}_{t,q}^{(I)}(f)$ has $\tilde{\boldsymbol{\Psi}}_{t,q}^{(I)}(f)$ and $\boldsymbol{M}_{\Omega_{t,q}^{(I)}}$ as frontal slabs.

C.3.3.3 Hybrid Training

In practice, one expects to have real data, but only in a limited amount. It makes sense to apply the notion of *transfer learning* [22, Ch. 15] as follows: first, learn an initial parameter vector $\hat{\boldsymbol{w}}$ by solving (C.5) with synthetic data. Second, solve (C.6) with real data, but using $\hat{\boldsymbol{w}}$ as initialization for the optimization algorithm. The impact of choosing this initialization is that the result of solving (C.6) in the second step will be closer to a "better" local optimum than if a worse initialization were adopted.

C.3.4 Deep Completion Autoencoders

This section proposes a deep neural network architecture based on *convolutional autoencoders* [29].

A (conventional) autoencoder [22, Ch. 12] is a neural network $p_{\boldsymbol{w}}$ composed of two parts, an encoder $\epsilon_{\boldsymbol{w}}$ and a decoder $\delta_{\boldsymbol{w}}$, which satisfy $p_{\boldsymbol{w}}(\boldsymbol{\Phi}) = \delta_{\boldsymbol{w}}(\epsilon_{\boldsymbol{w}}(\boldsymbol{\Phi}))\ \forall \boldsymbol{\Phi}$. The output of the encoder $\boldsymbol{\lambda} {:=} \epsilon_{\boldsymbol{w}}(\boldsymbol{\Phi}) \in \mathbb{R}^{N_\lambda}$ is referred to as the *code* or vector of *latent variables* and is of a typically much lower dimension than the input $\boldsymbol{\Phi}$. An autoencoder is trained so

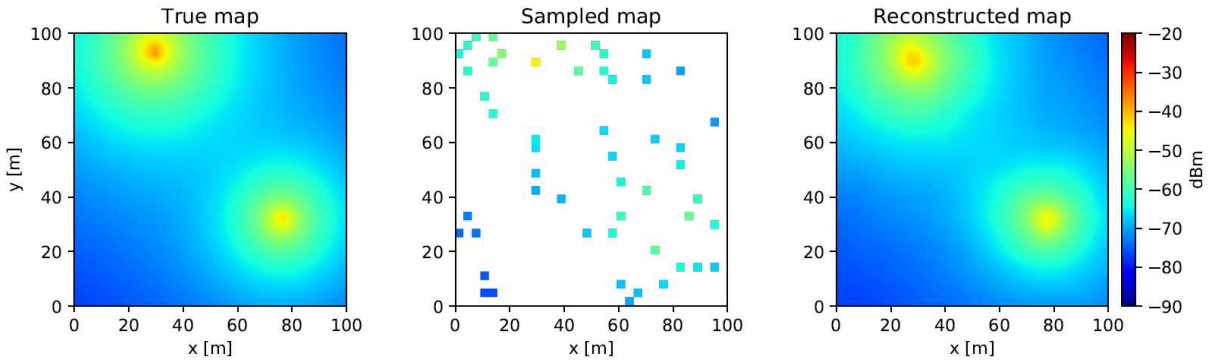

Figure C.2: Estimation with $N_\lambda = 4$ latent variables: (left) true map, (middle) sampled map portraying grid points $\{\boldsymbol{\xi}_{i,j}\}$ with $|\mathcal{A}_{i,j}| > 0$, and (right) estimated map.

that $\delta_{\boldsymbol{w}}(\epsilon_{\boldsymbol{w}}(\boldsymbol{\Phi})) \approx \boldsymbol{\Phi}\ \forall \boldsymbol{\Phi}$, which forces the encoder to compress the information in $\boldsymbol{\Phi}$ into the N_λ variables in $\boldsymbol{\lambda}$. The selection of N_λ will be addressed later.

A *completion* autoencoder adheres to the same principles as conventional autoencoders except for the fact that the encoder must determine the latent variables from a subset of the entries of the input. If a mask is used, the transfer function must satisfy $\boldsymbol{\Phi} \approx \delta_{\boldsymbol{w}}(\epsilon_{\boldsymbol{w}}(\mathcal{P}_\Omega(\boldsymbol{\Phi}), \boldsymbol{M}_\Omega))\ \forall \boldsymbol{\Phi}$ and for a sampling set Ω that preserves sufficient information for reconstruction. If Ω does not satisfy this requirement, then reconstructing $\boldsymbol{\Phi}$ is impossible regardless of the technique used. In the application at hand and with the notation introduced in previous sections, the above expression becomes $\tilde{\boldsymbol{\Psi}} \approx \delta_{\boldsymbol{w}}(\epsilon_{\boldsymbol{w}}(\check{\boldsymbol{\Psi}}))$.

As indicated earlier, autoencoders are useful only when most of the information in the input can be condensed in N_λ variables, i.e., when the possible inputs lie close to a manifold of dimension N_λ. To see that this is the case in spectrum cartography, an illustrating toy example is presented next. Suppose that there are two sources, each one with a fixed (yet possible different) power, that can be placed at arbitrary positions in $\mathcal{X}$ and suppose that propagation occurs in free space. All possible spectrum maps in this setup are defined by $N_\lambda = 4$ quantities, which correspond to the x and y coordinates of the two sources. Fig. C.2 illustrates this effect, where the left panel of Fig. C.2 depicts a true map $\boldsymbol{\Psi}$ and the right panel shows its estimate using a completion autoencoder with $N_\lambda = 4$. The quality of the estimate clearly supports the aforementioned manifold hypothesis. Details about the network and simulation setup are provided in Sec. C.4. In a real-world scenario, there may be more than two sources, their transmit power may not always be the same, and there are shadowing effects, which means that $N_\lambda \geq 4$ will be required.

Since space limitations prevent us from detailing every design decision, the rest of this section will be confined to outline the main aspects of the architecture developed in this work and summarized in Fig. C.3.

The encoder mainly comprises convolutional and pooling layers. The motivation for convolutional layers is three-fold: (i) relative to fully connected layers, they severely reduce the number of parameters to train and, consequently, the amount of data required. Despite this drastic reduction, (ii) convolutional layers are still capable of exploiting the spatial structure of maps and (iii) they result in shift-invariant transfer functions, a desirable property in the application at hand since moving the sources in a certain direction

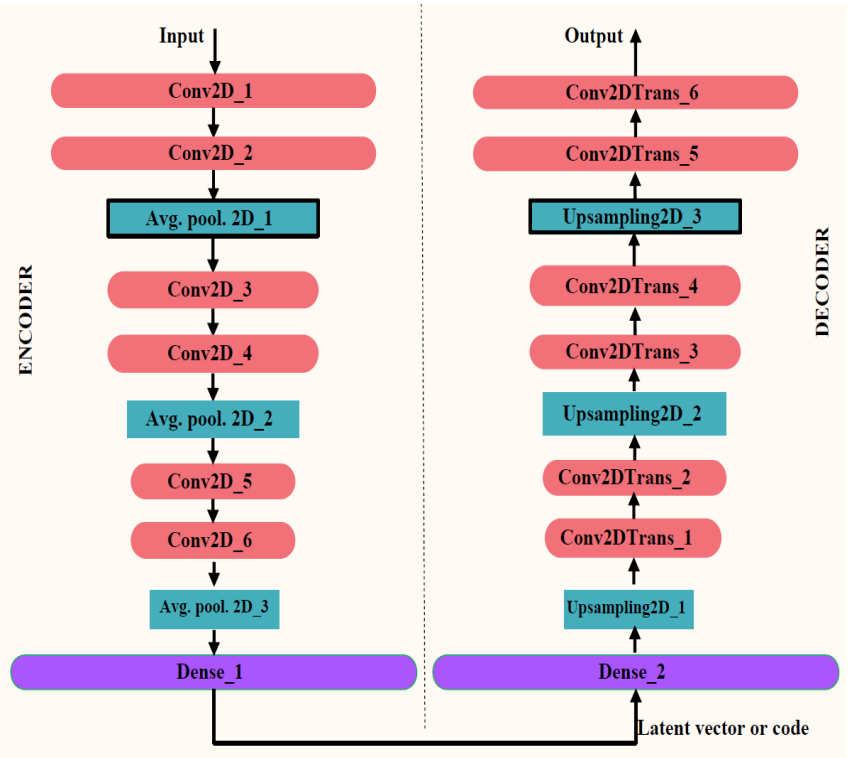

Figure C.3: Autoencoder architecture.

Table C.1: Parameters of the proposed network.

Layers	Parameters
Conv2D/ Conv2DTranspose	Kernel size = 3×3, stride = 1, activation = PLReLU, 64 filters
AveragePooling2D	Pool size = 2, stride = 2
Upsampling2D	Up-sampling factor = 2, bilinear interpolation
Dense	64 neurons (encoder), 1024 neurons (decoder)

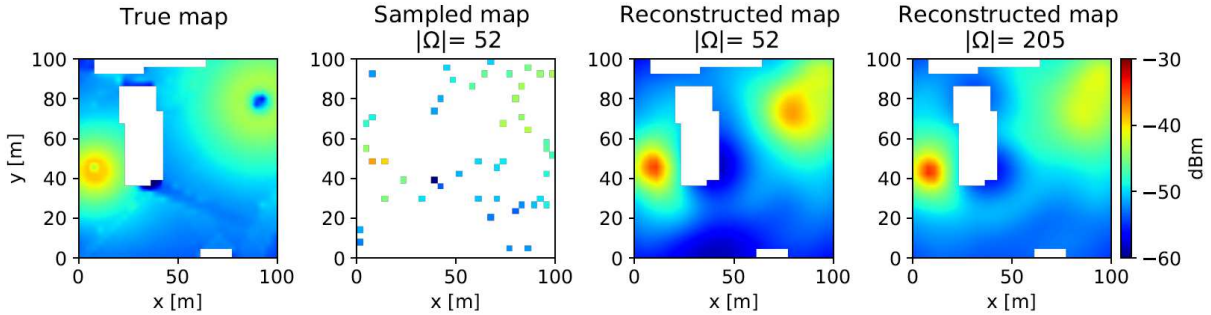

Figure C.4: Power map estimate with the proposed neural network. (left): true map, (center left): sampled map portraying the locations of the grid points $\{\boldsymbol{\xi}_{i,j}\}$ where $|\mathcal{A}_{i,j}| > 0$; (center right) and (right): estimated maps. White areas represent buildings.

must be corresponded by the same movement in the estimated map. These layers compute

$$[\boldsymbol{\Phi}^{(O)}]_{i,j,c_{\text{out}}} = \sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{u=-k}^{k} \sum_{v=-k}^{k} [\boldsymbol{F}_{c_{\text{out}}}]_{u,v,c_{\text{in}}} [\boldsymbol{\Phi}^{(I)}]_{i-u,j-v,c_{\text{in}}},$$

where $\mathbf{\Phi}^{(O)}$ is the output tensor, $\mathbf{\Phi}^{(I)}$ is the input tensor, and $\boldsymbol{F}_{c_{\text{out}}}$ is the c_{out}-th filter (or kernel), which is of size $2k+1 \times 2k+1$. Layer indices were omitted in order not to overload notation. The activation functions used here are parametric *leaky rectified linear units* (PLReLUs) [30] whose leaky parameter is also trained.

On the other hand, *average pooling* layers down-sample the outputs of convolutional layers, thereby condensing the information gradually in fewer features. Additionally, pooling features are approximately shift invariant as well [22, Ch. 9].

The last layer of the encoder is *fully-connected*. Since the previous layers were constrained to be convolutional or pooling layers, a final fully-connected layer is included in the encoder so that the latent variables can capture arbitrary relations among the shift invariant features obtained by the output of the second-to-last layer.

As usual in autoencoders, the decoder follows a "reverse" architecture relative to the encoder. Wherever the encoder has a convolutional layer, the decoder has a corresponding *convolution transpose* layer [31], sometimes called "deconvolutional" layer. Likewise, the pooling layers of the encoder are matched with *up-sampling* layers, which use bilinear interpolation in the architectures that we investigated. Finally, the fully connected layer of the encoder is paired with a fully connected layer in the decoder. The overall network architecture is summarized in Fig. C.3 and Table C.1.

C.4 Numerical Experiments

This section validates the proposed framework and network architecture through numerical experiments. Due to space limitations, the focus is on the most fundamental cartographic aspects, where the main novelty resides. Thus, $\mathcal{F}$ is set to the singleton $\mathcal{F} = \{900 \text{ MHz}\}$. $\mathcal{X}$ is a square area of side 100 m, discretized into a grid with $N_y = N_x = 32$. The two considered transmitters have height 1.5 m and transmit power 11 and 7 dBm over a bandwidth of 5 MHz.

Two classes of maps are generated. First, $T = 4 \cdot 10^5$ maps are obtained where the two transmitters are placed uniformly at random and where propagation adheres to the Gudmundson model [32] with pathloss exponent 3, gain at unit distance -30 dB, and shadowing correlation $\mathrm{E}\{H_l(\boldsymbol{x}_1, f)H_l(\boldsymbol{x}_2, f)\} = \sigma^2_{\text{sh}} 0.95^{||\boldsymbol{x}_1 - \boldsymbol{x}_2||}$ with $\sigma^2_{\text{sh}} = 10$ dB^2. Sensors are distributed uniformly at random without replacement across the grid points. A separate set of maps is generated using Remcom's Wireless InSite software in an urban scenario. Sensors are distributed uniformly at random without replacement across the grid points that lie on the streets. To better observe the impact of propagation phenomena, $\upsilon(\boldsymbol{x}, f)$ is set to 0. Each measurement $\tilde{\Psi}(\boldsymbol{\xi}_{i,j}, f)$ is obtained by adding zero-mean Gaussian noise with standard deviation 1 dB to $\Psi(\boldsymbol{\xi}_{i,j}, f),\ (i,j) \in \Omega$.

The network proposed in Sec. C.3.4 is implemented in TensorFlow and trained using the ADAM solver with learning rate 10^{-4}. Due to lack of space, only one training approach can be analyzed, in this case (C.5) with $\{(\check{\mathbf{\Psi}}_t, \mathbf{\Psi}_t)\}_{t=1}^T$ the Gudmundson data set. The algorithm is compared against the state-of-the-art competitors described next, whose parameters were tuned to approximately optimize their performance in the second experiment. (i) The kriging algorithm in [1] with regularization parameter 10^{-5} and Gaussian

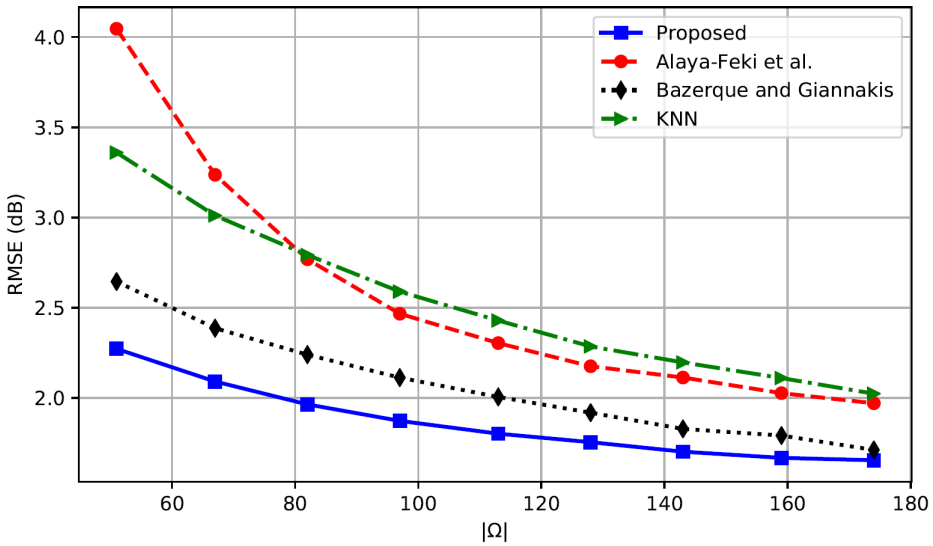

Figure C.5: Comparison with state-of-the-art alternatives. Even though the parameters of the competing algorithms were tuned for this specific experiment, the proposed network offers a markedly better performance.

radial basis functions with parameter $\sigma_K{:=}3\sqrt{\Delta_y N_y \Delta_x N_x/|\Omega|}$, which is approximately 3 times the mean distance between two points at which measurements have been collected. (ii) The multikernel algorithm in [17] with 20 Laplacian kernels with parameter uniformly spaced between $[0.1\sigma_K, \sigma_K]$ and regularization parameter 10^{-4}. As a benchmark, (iii) the K-nearest neighbors (KNN) algorithm with $K = 5$ is also shown.

The first experiment shows an estimated map using the proposed algorithm. The first panel of Fig. C.4 depicts the true map, which was generated using the Remcom data set. The second panel shows $\tilde{\mathbf{\Psi}}$ whereas the third and fourth show map estimates using different numbers of measurements. Observe that with just $|\Omega| = 52$ measurements, the estimate is already of a high quality. Note that details due to diffraction or the directivity of the antennas are not reconstructed because the Gudmundson model used to train the network does not capture them and therefore the network did not learn these features. This illustrates the need for training over data sets that model the reality as close as possible.

The second experiment compares the root mean square error

$$\text{RMSE} = \sqrt{\frac{\mathbb{E}\{||\mathbf{\Psi} - \hat{\mathbf{\Psi}}||_F^2\}}{N_y N_x}},$$

of the aforementioned algorithms, where $\mathbf{\Psi}$ is the true map, drawn at random from the Gudmundson data set, $\hat{\mathbf{\Psi}}$ is the estimated map, and $\mathbb{E}\{\cdot\}$ denotes expectation over maps, noise, and sensor locations. From Fig. C.5, the proposed scheme performs approximately a 20 % better than the next competing alternative. The parameters of the competing algorithms were tuned for this specific experiment, so their performance as in Fig. C.5 is optimistic. In practice one must expect a greater performance gap.

C.5 Conclusions

Learning propagation features from data yields spectrum cartography algorithms that require fewer measurements to attain a target performance. Deep neural networks can bring this idea into practice and offer a performance that beats the state-of-the-art. Future work will design more sophisticated network architectures relying on larger data sets.

References

[1] A. Alaya-Feki, S. B. Jemaa, B. Sayrac, P. Houze, and E. Moulines, "Informed spectrum usage in cognitive radio networks: Interference cartography," in *Proc. IEEE Int. Symp. Personal, Indoor Mobile Radio Commun.*, Cannes, France, Sep. 2008, pp. 1–5.

[2] J.-A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1847–1862, Mar. 2010.

[3] B. A. Jayawickrama, E. Dutkiewicz, I. Oppermann, G. Fang, and J. Ding, "Improved performance of spectrum cartography based on compressive sensing in cognitive radio networks," in *Proc. IEEE Int. Commun. Conf.*, Budapest, Hungary, Jun. 2013, pp. 5657–5661.

[4] N. Patwari and P. Agrawal, "Effects of correlated shadowing: Connectivity, localization, and RF tomography," in *Int. Conf. Info. Process. Sensor Networks*, St. Louis, MO, Apr. 2008, pp. 82–93.

[5] D. Romero, Donghoon Lee, and G. B. Giannakis, "Blind radio tomography," *IEEE Trans. Signal Process.*, vol. 66, no. 8, pp. 2055–2069, 2018.

[6] S. Grimoud, S. B. Jemaa, B. Sayrac, and E. Moulines, "A REM enabled soft frequency reuse scheme," in *Proc. IEEE Global Commun. Conf.*, Miami, FL, Dec. 2010, pp. 819–823.

[7] E. Dall'Anese, S.-J. Kim, G. B. Giannakis, and S. Pupolin, "Power control for cognitive radio networks under channel uncertainty," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3541–3551, Aug. 2011.

[8] D. Romero, S-J. Kim, G. B. Giannakis, and R. López-Valcarce, "Learning power spectrum maps from quantized power measurements," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2547–2560, May 2017.

[9] G. Boccolini, G. Hernandez-Penaloza, and B. Beferull-Lozano, "Wireless sensor network for spectrum cartography based on kriging interpolation," in *Proc. IEEE Int. Symp. Personal, Indoor Mobile Radio Commun.*, Sydney, NSW, Nov. 2012, pp. 1565–1570.

[10] S.-J. Kim, N. Jain, G. B. Giannakis, and P. Forero, "Joint link learning and cognitive radio sensing," in *Proc. Asilomar Conf. Signal, Syst., Comput.*, Pacific Grove, CA, Nov. 2011, pp. 1415 –1419.

[11] S.-J. Kim and G. B. Giannakis, "Cognitive radio spectrum prediction using dictionary learning," in *Proc. IEEE Global Commun. Conf.*, Atlanta, GA, Dec. 2013, pp. 3206 – 3211.

[12] D.-H. Huang, S.-H. Wu, W.-R. Wu, and P.-H. Wang, "Cooperative radio source positioning and power map reconstruction: A sparse Bayesian learning approach," *IEEE Trans. Veh. Technol.*, vol. 64, no. 6, pp. 2318–2332, Aug. 2014.

[13] G. Ding, J. Wang, Q. Wu, Y.-D. Yao, F. Song, and T. A Tsiftsis, "Cellular-base-station-assisted device-to-device communications in TV white space," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 107–121, Jul. 2016.

[14] M. Hamid and B. Beferull-Lozano, "Non-parametric spectrum cartography using adaptive radial basis functions," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, New Orleans, LA, Mar. 2017, pp. 3599–3603.

[15] Y. Teganya, D. Romero, L. M. Lopez-Ramos, and B. Beferull-Lozano, "Location-free spectrum cartography," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4013–4026, Aug. 2019.

[16] J.-A. Bazerque, G. Mateos, and G. B. Giannakis, "Group-lasso on splines for spectrum cartography," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4648–4663, Oct. 2011.

[17] J.-A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via kernel-based learning," *IEEE Signal Process. Mag.*, vol. 28, no. 30, pp. 112–125, Jul. 2013.

[18] S.-J. Kim, E. Dall'Anese, and G. B. Giannakis, "Cooperative spectrum sensing for cognitive radios using Kriged Kalman filtering," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 1, pp. 24–36, Jun. 2010.

[19] D. Lee, D. Berberidis, and G. B. Giannakis, "Adaptive Bayesian channel gain cartography," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Calgary, Canada, Apr. 2018, pp. 3555–3558.

[20] P. Stoica and R. L. Moses, "Spectral analysis of signals," 2005.

[21] D. P. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*, John Wiley & Sons, Inc., 2001.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[23] D. Romero, D. Lee, and G. B. Giannakis, "Blind radio tomography," *IEEE Trans. Signal Process.*, vol. 66, no. 8, pp. 2055–2069, Apr. 2018.

[24] B. R. Hamilton, X. Ma, R. J. Baxley, and S. M. Matechik, "Propagation modeling for radio frequency tomography in wireless networks," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 1, pp. 55–65, Feb. 2014.

[25] J. Fan and T. Chow, "Deep learning based matrix completion," *Neurocomputing*, vol. 266, pp. 540–549, Nov. 2017.

[26] V. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, 2007.

[27] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 107, Jul. 2017.

[28] M. C Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of communication systems: modeling, methodology and techniques*, Springer Science & Business Media, 2006.

[29] M. Ribeiro, A. E. Lazzaretti, and H. S. Lopes, "A study of deep convolutional auto-encoders for anomaly detection in videos," *Pattern Recognition Letters*, vol. 105, pp. 13–22, Apr. 2018.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vision*, Washington, DC, Dec. 2015, pp. 1026–1034.

[31] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[32] M. Gudmundson, "Correlation model for shadow fading in mobile radio systems," *Electron. Letters*, vol. 27, no. 23, pp. 2145–2146, Nov. 1991.