RF Transmitter Fingerprinting Exploiting Spatio-temporal Properties in Raw Signal Data

Debashri Roy*, Tathagata Mukherjee†, Mainak Chatterjee*, Eduardo Pasiliao3

* Computer Science, University of Central Florida, Orlando, FL 32826, {debashri, mainak}@cs.ucf.edu

† Computer Science, University of Alabama, Huntsville, AL 35899, {tm0130}@uah.edu

‡ Munitions Directorate, Air Force Research Laboratory, Eglin AFB, FL, 32542, {eduardo.pasiliao}@us.af.mil

***Abstract*—The recent advances of wireless technologies in RF environments coupled with large scale usage of such technologies has warranted more autonomous deployments of wireless systems. Machine learning techniques, that include recurrent structures, have shown promise in creating such autonomous deployments using the idea of Radio Frequency Machine Learning (RFML). In large scale autonomous deployments of wireless communication networks, the signals received from one component play a crucial role in the decision making process of other components. In order to efficiently implement such systems each component of the network should be uniquely identifiable. In this paper we propose a transmitter fingerprinting technique for radio device identification using recurrent structures, by exploiting the temporal property of the received radio signal. We design and implement three recurrent neural networks (RNNs) using different types of cell models: (i) long short term memory (LSTM); (ii) gated recurrent unit (GRU) and (iii) convolutional long short term memory (ConvLSTM), for this task. We program 8 universal software radio peripheral (USRP) software defined radios (SDRs) as transmitters and collect over-the-air raw inphase (I) and quadrature (Q) (I/Q) *time series* data from them using a DVB-T RTL-SDR receiver, in a laboratory setting. We exploit both the temporal variations as well as the inherent spatial dependencies in the collected I/Q time series data, to learn unique feature representations and use these as "fingerprints" for identifying the transmitters. Experimental results reveal that the RNNs with LSTM, GRU, and ConvLSTM cells are able to correctly distinguish between the 8 transmitters with 92%, 95.3%, 97.2% accuracy respectively.**

Keywords: RF fingerprinting, recurrent neural network, supervised learning, software defined radios.

I. Introduction

We are living in a world where the distances are shrinking every day, thanks to an explosion in the use of connected devices. The ubiquitous usage of wirelessly connected Internet-of-Things (IoT) [1] along with the deployment of wireless autonomous systems has ushered in a new era of industrial scale deployment of radio frequency (RF) devices. This prevalence of large scale peer-to-peer communication and the nature of the underlying ubiquitous network brings forth the challenge of accurately identifying a RF transmitter. Every device that is part of a large network needs to be able to identify its peers with high confidence in order to set up secure communication channels. One of the ways in which this is done is through the interchange of "keys" [2] for host identification. However, such schemes are prone to breaches by malicious agents [3] because often the actual implementations of such systems are not cryptographically sound. In order to get around the problem of faulty implementations, one can use the transmitter's intrinsic characteristics to create a "fingerprint" that can be used by a transmitter identification system. Every transmitter, no matter how similar, has intrinsic characteristics because of the imperfections in its underlying components such as amplifiers, filters, frequency mixers as well as the physical properties of the transmitting antenna; these characteristics are unique to a specific transmitter. The inaccuracies present in the manufacturing process and the idiosyncrasies of the hardware circuitry also contribute to the spatial and temporal characteristics of the signal transmitted through a particular device.

This inherent heterogeneity can be exploited to create unique identifiers for the transmitters. One such property is the imbalance in the Inphase (I) and Quadrature (Q) phase components of the signal (I/Q data). However, because of the sheer number of the transmitters involved, manually "fingerprinting" each and every transmitter is not a feasible task [4]. Thus, in order to build such a system, there needs to be an "automatic" method of extracting the transmitter characteristics and using the resulting "fingerprint" for the differentiation process. One way of achieving this is by learning the representation of the transmitter in an appropriate "feature space" that has enough discriminating capability so as to be able to differentiate between "apparently identical" transmitters.

Among the various approaches that can be used to discern this feature space, deep learning (DL [5]) based methods provide an efficient and automatic way of learning and characterizing the feature space. They are able to learn and analyze the inherent properties of large deployments and use it to predict and characterize the associated parameters for the task of automatic feature learning for classification (or regression). Deep neural networks have been shown to be effective for automatically learning discriminating features from data for various tasks [6]. With proper choice of the neural network architecture and associated parameters, they can compute arbitrarily good function approximations [7]. Since the task of classification is equivalent to learning the decision boundary, neural networks were a natural candidate for a learning machine. Coupled with their prior success in learning from signal data, they were the natural choice for implementing our system.

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00023

Neural networks have previously been used for transmitter identification [8]–[10] and are particularly attractive since they can generate accurate models without knowledge of the *apriori* data distribution. Neural networks have been shown to be able to predict modulation techniques [8] and identify transmitters [10] by only considering the spatial correlations within the actual [9] or synthetic RF data [11]. It is to be noted that all prior works have only exploited the spatial correlation of the signal data, though a continuous signal can be represented as a time series, having both temporal and spatial properties [12].

Inspired by the success of deep learning systems for the task of characterizing RF environments [13] and the successful use of recurrent neural networks (RNN) for the task of analyzing time series data [14], we propose to use deep recurrent structures for learning transmitter "fingerprints" for the task of transmitter identification. Recurrent Neural Networks (RNNs) [6] have been shown to be useful for capturing and exploiting the temporal correlations of time series data. There are a few variants of recurrent neural networks: (i) Long Short-Term Memory (LSTM) [15], (ii) Gated Recurrent Unit (GRU) [16], and (iii) Convolutional Long Short-Term Memory (ConvLSTM) [17]. All these variants are designed to learn the long term temporal dependencies and are capable of avoiding the "vanishing" or "exploding" gradient problems [18].

In this paper, we use these variants of recurrent neural networks with time series of I/Q data to identify different transmitters. The main contributions of this paper are:

1) We exploit the temporal properties of I/Q data by using a supervised learning approach for transmitter identification using recurrent neural networks. We use two approaches: first, we exploit only the temporal property and then we exploit the spatio-temporal property. We use RNNs with LSTM and GRU cells for the first approach while we use a convLSTM model for the latter. Although transmitter fingerprinting has been studied before, to the best of our knowledge this is the first work which leverages the spatio-temporal property of the over-the-air signal data for this task.
2) To examine the performance of the proposed networks, we test them on an indoor testbed. We transmit raw signal data from 8 universal software radio peripheral (USRP) B210s [19] and collect over-the-air signals using a RTL-SDR [20]. We use the I/Q values from each USRP for fingerprinting the corresponding transmitter.
3) The novelty of this work lies in accurately modeling and implementing different types of RNNs to build a robust transmitter fingerprinting system using over-the-air signal data, by exploiting spatio-temporal correlations.

The rest of the paper is organized as follows: in the next section, we present a survey of existing machine learning based transmitter identification techniques. In section III, we propose different RNN models. In section IV, we present the testbed setup and experiments that we conduct to evaluate the proposed models. We present the experimental results in section V. Conclusions are drawn in the last section.

II. Related Works

Recurrent neural networks [6] have been used extensively for modeling *temporal data* such as speech [14]. There is limited amount of work that recognizes the potential of using recurrent structures in the RF domain and in general the use of deep learning in the RF domain has been limited in the past with only a few applications in recent times [8], [13].

In [8], the authors have demonstrated the use of neural networks for modulation detection. Apart from the results, an interesting aspect of the work is the way I/Q values were used as input to the neural network. More precisely, given N I/Q values, the authors used a vector of size $2N$ as an input to the neural network, effectively using the I and Q components as a tuple representing a point in the complex plane. This representation proves to be useful for using the I/Q data in different learning models.

In [21], O'Shea et. al. presented a recurrent neural network that extracted high level protocol information from the low level physical layer representation for the task of classification. A radio anomaly detection technique was presented in [22], where the authors used a LSTM based RNN as a time series predictor using the error component to detect anomaly from real signals. Another application of RNN was proposed in [23], where the authors used a deep recurrent neural network to learn the time-varying probability distribution of received powers on a channel and used the same to predict the suitability of sharing that channel with other users. A method for modulation classification was proposed in [24] for a distributed wireless spectrum sensing network. The authors proposed a recurrent neural network using long short term memory (LSTM) cell, yielding 90% accuracy on a synthetic dataset [25]. Bai et. al. proposed an end-to-end RF fingerprinting [26] method using two RNNs in order to learn the spatial or temporal pattern. Both simulated and real-world data was used to improve the positioning accuracy and robustness of moving RF devices.

For the task of transmitter classification, there are a few traditional methods that use manual feature engineering and leverage different radio attributes like transients, or spurious modulations to create discriminating feature sets. A transient signal is transmitted when a transmitter is powered up or powered down. During this short period (typically a few micro seconds) capacitive loads charge or discharge. Different classification approaches using transient based recognition was proposed in [27]–[29]. A different approach for transmitter fingerprinting and classification was proposed in [30], where the authors classified FM radio transmitters based on unique stray features extracted from spurious modulation characteristics. However, none of these methods address the problem of providing an end-to-end solution using raw signal data for transmitter identification using automatically extracted "fingerprints".

III. Proposed RNN Models for Classification

In order to estimate the noise in a RF channel, the system needs to "listen" to the underlying signal for sometime and "remember" the same. Previously, neural networks lacked this capability when used in the context of temporal data. Another issue with using neural networks with temporal data was the problem of *vanishing gradients*, when trying to use *back propagation*. Both these problems were solved by the introduction of Recurrent Neural Networks (RNN) [5].

a) Formulation of temporal property of RF data: Given T training samples (for T timestamps) where each training sample is of size of M and consists of a vector of tuples of the form $(I,Q) \in \mathcal{C}$ representing a number in the complex plane, we represent a single sample as $x_t = [[(I,Q)_i]^t; i = 1,2,\cdots,M] \in \mathcal{C}^M$ for each timestamp $t = 1,2,\cdots,T$, and we use it as an input to the neural network. We use a sample size (M) of 1024 as a default. We want to find the probability of the input vector for next time step (x_{t+1}) to belong to class C_k, where $k \in 1,2,\ldots,K$, K being the number of classes. The probability $P(C_k|x_{t+1})$ can be written as

$$P(C_k|x_{t+1}) = \frac{P(x_t|C_k)P(C_k)}{P(x_t x_{t+1})} \tag{1}$$

where $(P(x_t|C_k))$ is the conditional probability of x_t given the class C_k and $(P(x_t x_{x+1}))$ is the probability of x_t and x_{t+1} occurring *in order.*

A. Long Short Term Memory (LSTM) Cell Model

Though LSTM cells can be modeled and designed in various ways depending on the need, we use the cells as shown in Fig. 1. In one LSTM cell, there are (i) three types of gates: input (i), forget (f), and output (o); and (ii) a state update of internal cell memory. The most interesting part of the LSTM cell is the "forget" gate, which at time t is denoted by f_t. The forget gates decide whether to keep a cell state memory (c_t) or not. The forget gates are designed as per the equation (2) on the input value x_t at time t and output (h_{t-1}) at time $(t-1)$.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2}$$

Note that W_{xf} and b_f represent the associated weight and bias respectively, between input (x) and forget gate (f) and σ denotes the *sigmoid* activation function. Once f_t determines which memories to forget, the input gates (i_t) decides which cell states ($\widetilde{c_t}$) to update as per equations (3) and (4).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{3}$$

$$\widetilde{c_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c_{t-1}}) \tag{4}$$

In equation (5), the old cell state (c_{t-1}) is updated to the new cell state (c_t) using forget gates (f_t) and input gates (i_t).

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c_t} \tag{5}$$

Here $\circ$ is the Hadamard product. Finally, we filter the output values through the output gates (o_t) based on the cell states (c_t) as per equations (6) and (7).

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_t + b_o) \tag{6}$$

$$h_t = o_t \circ tanh(c_t) \tag{7}$$

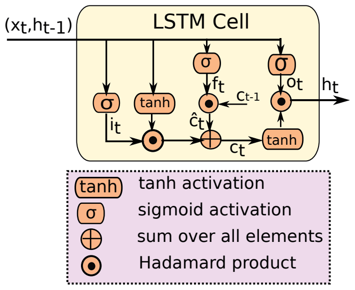


Fig. 1. LSTM Cell Architecture Used in the RNN Model

B. Gated Recurrent Unit (GRU) Model

The main drawback of using LSTM cells is the need for additional memory. GRUs [16] have one less gate for the same purpose, thus having a reduced memory and CPU footprint. The GRU cells control the flow of information just like the LSTM cells, but without the need for a memory unit. It simply exposes the full hidden content without any control. It has a "reset gate" (z_t), an "update gate" (r_t), and a cell state memory (c_t) as shown in Fig. 2. The reset gates determine whether to combine the new input with a cell state memory (c_t) or not. The update gate decides how much of c_t to retain. The equations (8), (9), (10), and (11), related to the different gates and states are given below.

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{8}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{9}$$

$$c_t = tanh(W_{xc}x_t + W_{hc}(r_t \circ h_{t-1})) \tag{10}$$

$$h_t = (1 - z_t) \circ c_t + z_t \circ h_{t-1} \tag{11}$$

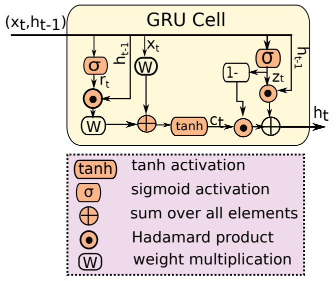


Fig. 2. GRU Cell Architecture Used in the RNN Model

C. Convolutional LSTM Network Model

The recurrent neural networks with LSTM or GRU cells, do not consider the spatial information encoded in the the input-to-state or state-to-state transitions. To mitigate this problem, we use a convolution within the recurrent structure of the RNN. We first discuss the spatio-temporal property of RF data and then model a convolutional LSTM network to exploit the same.

1) Formulation of Spatio-temporal property for RF data: Suppose that a radio signal is represented as a time varying series over a spatial region using R rows and C columns. Here R represents the time varying nature of the signal and as such in our case it represents the total number of time stamps at which the signal was sampled (T in our case). C on the other hand represents the total number of features sampled at each time stamp (in our case its 2048 since there are 1024 features sampled each of dimension 2). Note that each cell corresponding to one value of R and one value of C represents a particular feature (I or Q) at a given point in time.

In order to capture the temporal property only, we use a sequence of vectors corresponding to different timestamps $1, 2, \cdots, t$ as $x_1, x_2, \cdots, x_t$. However, to capture both spatial and temporal properties, we introduce a new vector $\chi_{t,t+\gamma}$, which is formulated as: $\chi_{t,t+\gamma} = [x_t, x_{t+1}, \cdots, x_{t+\gamma-1}]$. So the vector $\chi_{t,t+\gamma}$ eventually preserves the spatial properties with an increment of γ in time. So, we get a sequence of new vectors $\chi_{1,\gamma}, \chi_{\gamma,2\gamma}, \cdots \chi_{t,t+\gamma}, \cdots, \chi_{t+(\beta-1)\gamma,t+\beta\gamma}$, where β is $\lfloor R/\gamma \rfloor$, and the goal is to create a model to classify them into one of the K classes (corresponding to the transmitters). We model the class-conditional densities given by $P(\chi_{t-\gamma,t}|C_k)$, where $k \in 1, \cdots, K$. We formulate the probability of the next γ-length sequence to be in class C_k as per equation 12. The marginal probability is modeled as $P(\chi_{t,t+\gamma})$.

$$P(C_k|\chi_{t,t+\gamma}) = \frac{P(\chi_{t-\gamma,t}|C_k)P(C_k)}{P(\chi_{t,t+\gamma})} \tag{12}$$

2) The Model: The cell model is similar to an LSTM cell, but the input transformations and recurrent transformations are both convolutional in nature [17]. We formulate the input values, cell state and hidden states as a 3-dimensional vector, where the first dimension is the number of measurements which varies with the time interval γ and the last two dimensions contain the spatial information (rows (R) and columns (C)). We represent these as: (i) the inputs: $\chi_{1,\gamma}, \chi_{\gamma,2\gamma}, \cdots \chi_{t,t+\gamma}, \cdots, \chi_{t+(\beta-1)\gamma,t+\beta\gamma}$ (previously stated); (ii) cell outputs: $\mathcal{C}_1, \cdots, \mathcal{C}_t$, and (iii) hidden states: $\mathcal{H}_1, \cdots, \mathcal{H}_t$. We represent the gates in a similar manner as in the LSTM model. The parameters t, i_t, f_t, o_t, W, b hold the same meaning as in section III-A. The key operations are defined in equations 13, 14, 15, 16, and 17. The probability of the next γ-sequence to be in a particular class (from equation 12) is used within the implementation and execution of the model.

$$i_t = \sigma(W_{xi}\chi_{t,t+\gamma} + W_{hi}\mathcal{H}_{t-1} + b_i) \tag{13}$$

$$f_t = \sigma(W_{xf}\chi_{t,t+\gamma} + W_{hf}\mathcal{H}_{t-1} + b_f) \tag{14}$$

$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t . \tanh(W_{xc}\chi_{t,t+\gamma} + W_{hc}\mathcal{H}_{t-1} + b_c) \tag{15}$$

$$o_t = \sigma(W_{xo}\chi_{t,t+\gamma} + W_{ho}\mathcal{H}_{t-1} + b_o) \tag{16}$$

$$\mathcal{H}_t = o_t \circ \tanh(\mathcal{C}_t) \tag{17}$$

IV. Testbed Evaluation

In order to validate the proposed models, we collected raw signal data from 8 different universal software radio peripheral (USRP) B210s [19]. We collected the data in an indoor lab environment with a signal-to-noise ratio of 30 dB, and used the dataset to distinguish between 4 or 8 transmitters, as mentioned in [10].

Parameters	Values
Transmitter Gain	45 dB
Transmitter Frequency	904 MHz (ISM)
Bandwidth	200 KHz
Sample Size	1024
Samples/Transmitter	40,000
# Transmitters	4 and 8

TABLE I
Transmission Configuration Parameters

A. Signal Generation and Data Collection

In order to evaluate our methods for learning the inherent spatio-temporal features of a transmitter, we used eight USRPs of the same type, namely B210 from Ettus Research [19], as transmitters. The signal generation and reception are shown in Fig. 3. We used GNURadio [31] to randomly generate signal and modulated the same with Quadrature Phase Shift Keying (QPSK). We programmed the USRP B210s to transmit the modulated signal over the air and sensed the same using a DVB-T dongle (RTL-SDR) [20]. We generated the entire dataset from "over-the-air" data as sensed by the RTL-SDR using the *rtlsdr* python library.

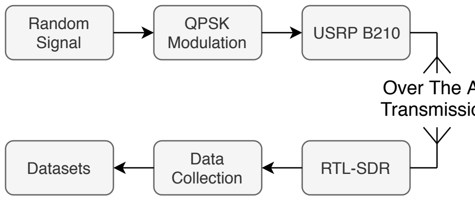


Fig. 3. Over the Air Signal Generation and Data Collection Technique

We collected I/Q signal data with a sample size of 1024 at each time stamp. Each data sample had 2048 entries consisting of the I and Q values for the 1024 samples. Note that a larger sample size would mean more training examples for the neural network. Our choice of 1024 samples was sufficient to capture the spatial-temporal properties while at the same time the training was not computationally intensive. We collected 40,000 training examples from each transmitter to avoid the data skewness problem observed in machine learning. The configuration parameters that were used are given in Table I. We collected two sets of data: (i) using 4 transmitters: 6.8 GB size, 160K rows and 2048 columns and (ii) using 8 transmitters: 13.45 GB size, 320K rows and 2048 columns. Note that we intend to make the dataset publicly available upon publication of the article.

B. Spatial Correlation in the Dataset

Correlation between data samples play a crucial role in the process of transmitter identification. We represent the I and Q values of each training sample at time (t) as: $[I_0Q_0I_1Q_1I_2Q_2I_3Q_3I_4Q_4 \cdots I_{1023}Q_{1023}]^t$. We used the QPSK modulation [32] which means that the spatial correlation should be between every fourth value, i.e., between I_0 and I_4, and Q_0 and Q_4. So we calculate the correlation coefficient of $I_0I_1I_2I_3$ and $I_4I_5I_6I_7$. Similarly, for $Q_0Q_1Q_2Q_3$ and $Q_4Q_5Q_6Q_7$. We take the average of all the correlation coefficients for each sample.

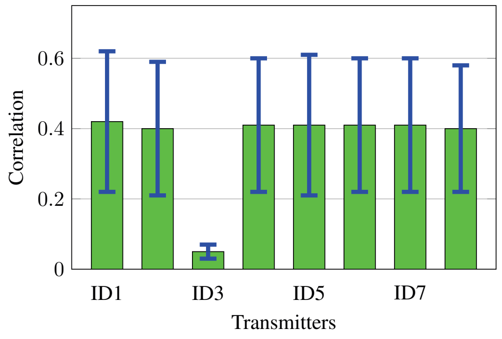


Fig. 4. Spatial Correlation in the Dataset

We use *numpy.corrcoef* for this purpose which uses Pearson product-moment correlation coefficients, denoted by r. The Pearson's method for a sample is given by:

$$r = \frac{\sum_{i=0}^{(M-1)} (I_i - \bar{I})(Q_i - \bar{Q})}{\sqrt{\sum_{i=1}^{(M-1)} (I_i - \bar{I})^2} \sqrt{\sum_{i=0}^{(M-1)} (Q_i - \bar{Q})^2}} \tag{18}$$

where, M is the sample size, I_i and Q_i are the sample values indexed with i. The sample mean is $\bar{I} = \frac{1}{M} \sum_{i=0}^{(M-1)} I_i$.

The spatial correlations of all the samples for the different transmitters are shown in Fig. 4. We observe that for most of the transmitters, the correlation is ∼0.42, with a standard deviation of ∼0.2. However, transmitter 3 exhibits minimal correlation between these samples, which implies that the spatial property of transmitter 3 is different from the other transmitters. As a result Transmitter 3 should be easily distinguishable from the others. This claim will be validated later in the experimental result section where we see 0% false positive and false negative for transmitter 3 for all the three proposed models. This observation gives us the motivation to exploit the spatial property as well as the temporal property for the collected *time-series* data.

C. Neural Network Libraries

There are many libraries available in *python* with support for different types of neural network and concurrent GPU architecture. We use *Keras* [33] as the frontend and *Tensorflow* [34] as the backend for our implementations. Keras is an overlay on the neural network primitives provided by Tensorflow [34] or Theano [35] and provides a customizable interface for quick deployment of complex neural networks. We also use *Numpy*, *Scipy*, and *Matplotlib* Python libraries.

D. Experimental Setup and Performance Metrics

We conducted the experiments on a Ryzen 8 Core system with 64 GB RAM, a GTX 1080 Ti GPU unit having 11 GB memory. During the training phase, we use data from each transmitter to train the neural network model. In order to test the resulting trained model, we use test data collected from one of the transmitters and present the same to the trained network. In general to measure the effectiveness of any learning algorithm, "accuracy" is used as the typical performance metric. However, accuracy can sometimes be misleading and incomplete when the data is skewed. For the task of classification, a confusion matrix overcomes this problem by showing how confused the learned model is on its predictions. It provides more insights on the performance by identifying not only the number of errors, but more importantly the types of errors.

V. Model Implementations and Results

In this section we discuss the implementation of each of the proposed recurrent neural networks. We train each network for transmitter classification with K classes. For the sake of robustness and statistical significance, we present the results for each model after averaging over several runs.

A. Implementation with LSTM Cells

As discussed earlier, the recurrent structure of the neural network can be used to exploit the temporal correlation in the data. To that end, we first implemented a recurrent neural network with LSTM cells and trained it on the collected dataset using the paradigm as shown in Fig. 5. We used two LSTM layers with 1024 and 256 units sequentially. We also used a *dropout* rate of 0.5 in between these two LSTM layers. Next we used two fully connected (*Dense*) layers with 512 and 256 nodes respectively. We apply a *dropout* rate of 0.2, and add *batch normalization* [36] on the output, finally passing it through a *Dense* layer having 8 nodes. We use *ReLU* [37] as the activation function for the LSTM layers and *tanh* [38] for the *Dense* layers. Lastly, we use *stochastic gradient descent* [38] based optimization with categorical cross-entropy training. Note that the neural network architecture was finalized over several iterations of experimentation with the data and we are only reporting the final architecture here. We achieved 97.17% and 92.00% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are shown in Figs. 6 and 7 respectively. Note that the number of nodes in the last layer is equal to the number of classes in the dataset. It is also to be noted that during the process of designing the RNN architecture, we also fine tuned the hyper-parameters based generalization ability of the current network (as determined by comparing the training and validation errors). We also limited the number of recurrent layers and fully connected layers for each model for faster training [39], since no significant increase in the validation accuracy was observed after increasing the number of layers.

The rows and columns of the confusion matrix correspond to the number of transmitters (classes) and the cell values show the recall or sensitivity and false negative rate for each of the transmitters. Note that recall or sensitivity represents the true positive rates for each of the prediction classes.

B. Implementation with GRU Cells

Next we implemented another variation of the RNN model using GRU cells for leveraging temporal correlation. We used

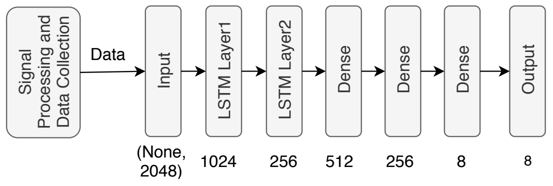
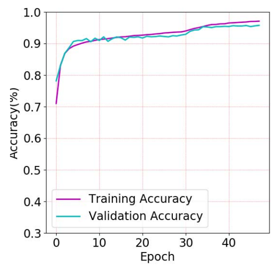
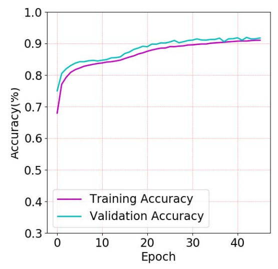


Fig. 5. RNN Implementation with LSTM Cells for Transmitter Classification



(a) 4 Transmitters



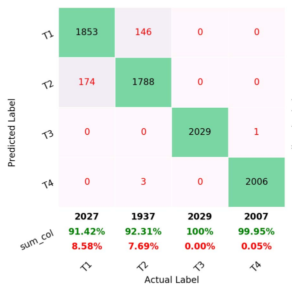
(b) 8 Transmitters

Fig. 6. Accuracy Plots for Transmitter Classification using LSTM Cells

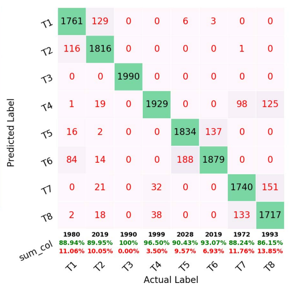
the same architecture as the LSTM implementation, presented in Fig. 8. The proposed GRU implementation needs fewer parameters than the LSTM model. A quantitative comparison is given in Section V-D. The only difference is that we use two GRU layers with 1024 and 256 units instead of using LSTM cells. We achieved 97.76% and 95.30% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are given in Figs. 9 and 10. The GRU implementation provided a slight improvement over the accuracy obtained using LSTM, for each run of the models, for both the datasets.

C. Implementation with ConvLSTM2D Cells

Finally, in order to exploit the spatio-temporal property of the signal data, we implemented another variation of the LSTM model with convolutional filters (transformations). The implemented architecture is shown in Fig. 11. *ConvLSTM2D* uses two dimensional convolutions for both input transformations and recurrent transformations. We first use two layers of *convLSTM2D* with 1024 and 256 filters respectively, and a *dropout* rate of 0.5 in between. We use kernel size of (2,2) and stride of (2,2) at each *ConvLSTM2D* layer. Next we add two fully connected (*Dense*) layers having 512 and 256 nodes respectively after *flattening* the convolutional output. *ReLU* [37], and *tanh* [38] activation functions are used for the *convLSTM2D* and *Dense* layers respectively. ADADELTA [40] with a learning rate of 10^{-4} and a decay rate of 0.9, is used as the optimizer with categorical cross-entropy training. We achieved 98.9% and 97.2% testing accuracy for 4 and 8 transmitters respectively. The accuracy plots and confusion matrices are given in Figs. 12 and 13 respectively. Being able to exploit the spatio-temporal correlation, ConvLSTM implementation provides improvement over the accuracies obtained using the LSTM and GRU models, for both the datasets.



(a) 4 Transmitters



(b) 8 Transmitters

Fig. 7. Confusion Matrices for Transmitter Classification using LSTM Cells

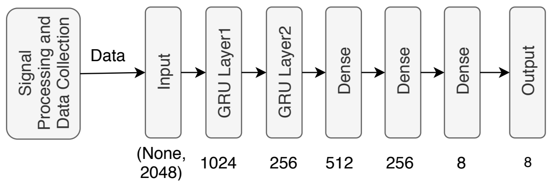
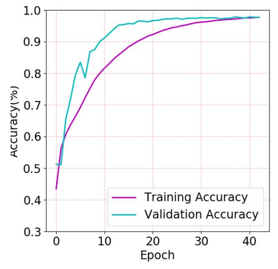
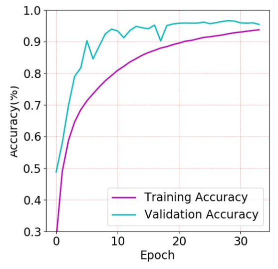


Fig. 8. RNN Implementation with GRU Cells for Transmitter Classification

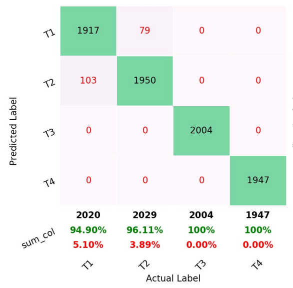


(a) 4 Transmitters

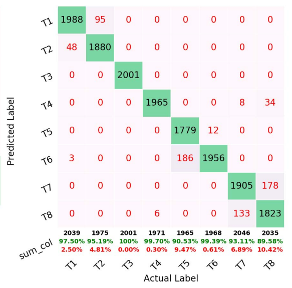


(b) 8 Transmitters

Fig. 9. Accuracy Plots for Transmitter Classification using GRU Cells



(a) 4 Transmitters



(b) 8 Transmitters

Fig. 10. Confusion Matrices for Transmitter Classification using GRU Cells

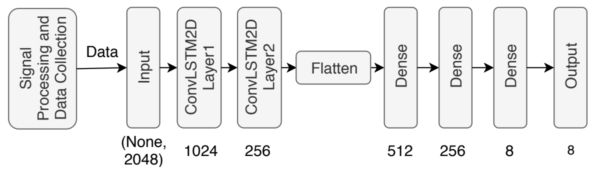
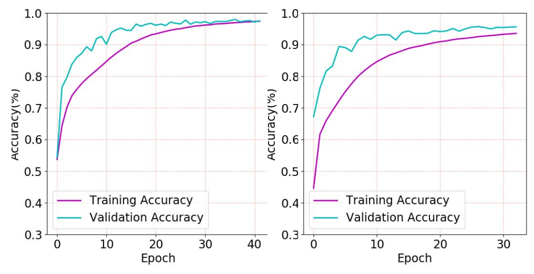


Fig. 11. RNN Implementation with ConvLSTM Cells for Transmitter Classification

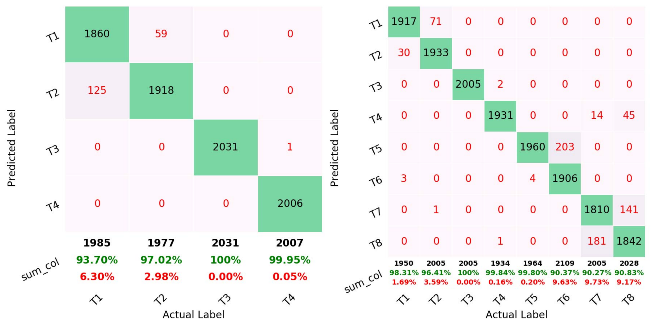
D. Comparisons of LSTM/GRU/ConvLSTM Implementations

We used 90%, 5%, and 5% of the data to train, validate, and test respectively. We ran each model for 50 epochs with early-stopping on the validation set. One epoch consists of a forward pass and a backward pass through the implemented



(a) 4 Transmitters (b) 8 Transmitters

Fig. 12. Accuracy Plots for Transmitter Classification using ConvLSTM Cells



(a) 4 Transmitters (b) 8 Transmitters

Fig. 13. Confusion Matrices for Transmitter Classification using ConvLSTM Cells

architecture for the entire dataset. The overall accuracy of the different implementations is shown in Table II. We find that the implementation of convolutional layers with recurrent structure (*ConvLSTM2D*) exhibit the best accuracy for transmitter classification, which clearly shows the advantage of using the spatio-temporal correlation present in the collected datasets. In Fig. 14, we present a better illustration of the achieved classification accuracies for the different implemented models.

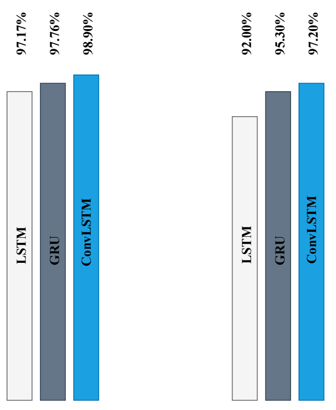
E. Comparisons of Proposed and Existing Approaches

Next we present two comparative studies of our proposed implementations with some existing techniques. We introduce a differential analysis of different RNN based implementations in the RF domain in Table III. Another comparative study for different transmitter classification techniques is shown in Table IV.

The "Inputs" column in both the tables refer to the type of inputs used for the methods under consideration. Table III shows a comparison of our ConvLSTM based RNN for transmitter classification with other RNN based implementations for separate tasks like modulation recognition and traffic sequence recognition. Table IV establishes the efficacy of our ConvLSTM based RNN model for the task of transmitter classification by comparing the accuracy with that obtained using other methods, for the same task. It is to be noted that all the other methods use expert crafted features as inputs ([27]–[30]), or work with synthetic datasets ([21], [24]). Our method, on the other hand achieves superior accuracy (97.2%) using features automatically learned from the raw signal data, thereby paving the way for real-time deployment of large scale transmitter identification systems.

#Trans	Models	#Parameters	Acc (%)
4	LSTM (6 layers)	14.2 M	97.17
4	GRU (6 layers)	10.7 M	97.76
4	ConvLSTM (6 layers)	14.2 M	98.90
8	LSTM (6 layers)	14.2 M	92.00
8	GRU (6 layers)	10.7 M	95.30
8	ConvLSTM (6 layers)	14.2 M	97.20

TABLE II
ACCURACY FOR DIFFERENT IMPLEMENTATIONS



(a) 4 Transmitters (b) 8 Transmitters

Fig. 14. Comparison of Testing Accuracies of Different Types of Recurrent Neural Networks

Approaches	Model	SNR (dB)	Acc (%)	Inputs
Traffic Sequence Recognition [21]	LSTM	20	31.2	Hybrid Real-synthetic Dataset
Automatic Modulation Classification [24]	LSTM	20	90	Synthetic Dataset [25]
Transmitter Classification (Ours)	ConvLSTM	30	97.2	Raw Signal

TABLE III
COMPARISON OF PROPOSED APPROACH WITH THE EXISTING RNN IMPLEMENTATIONS

It must be pointed out that the proposed RNN models can be a trained using raw signal data from any type of radio transmitter operating both in indoor as well as outdoor environments. We would also like to point out that though our data was collected in a lab environment, we had no control over the environment, there were other transmissions in progress, people were moving in and out of the lab and there was a lot of multi-path due to the location and design of the lab. Furthermore the power of the transmitters was low and hence this compounded the problem further. Given this, though we say that the data was collected in a lab environment, in reality it was an uncontrolled daily use environment reflective of our surroundings. Thus we can safely say that these methods will work in any real world deployment of large scale radio network. In summary,

- Exploiting temporal correlation only, recurrent neural networks yield 95-97% accuracy for transmitter classifi-

Approach	#Trans	SNR (dB)	Acc (%)	Inputs
Orthogonal Component Reconstruction (OCR) [30]	3	20	62 - 71	Spurious Modulation
Genetic Algorithm [27]	5	25	85-98	Transients
Multifractal Segmentation [28]	8	Not mentioned	92.5	Transients
k-NN [29]	8	30	97.2	Transients
Ours	8	30	97.04	Raw Signal

TABLE IV
COMPARISON OF THE OUR IMPLEMENTATION WITH THE EXISTING TRANSMITTER CLASSIFICATION APPROACHES

cation using LSTM or GRU cells. RNN implementation with GRU cells needs fewer parameters than LSTM cells as shown in Table II.

- Exploiting spatio-temporal correlation, the implementation of RNN using ConvLSTM2D cells provides better accuracy (97-98%) for transmitter classification, thus providing a potential tool for building automatic real world transmitter identification systems.
- We present a comparative study of the proposed spatio-temporal property based fingerprinting with the existing traditional and neural network based models. This clearly shows that the proposed model achieves the best accuracy compared to any of the existing methods for the task.

VI. Conclusion

In this paper, we proposed a robust transmitter identification technique by exploiting both the inherent spatial and temporal properties of RF signal data. We designed and implemented three different types of neural network models for this purpose. We collected over-the-air signal data from USRP B210s and used the same to train and validate our system. The RNN model using LSTM cells yields 92% testing accuracy leveraging only temporal property of the signal data. The one using GRU cells does the same with lower space requirement and yields a better testing accuracy of 95.30%. Finally, the RNN model with ConvLSTM cell achieves 97.20% testing accuracy for the same dataset leveraging both the temporal and spatial correlations. In the future we plan to use these methods for identification of actual infrastructure transmitters (for example FM, AM and GSM) in real world settings.

References

[1] R. H. Weber and R. Weber, *Internet of Things*. Springer, 2010, vol. 12.

[2] Y. B. Saied and A. Olivereau, "D-HIP: A distributed key exchange scheme for HIP-based Internet of Things," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012, pp. 1–7.

[3] M. Stanislav and T. Beardsley, "Hacking IoT: A case study on baby monitor exposures and vulnerabilities," *Rapid 7*, 2015.

[4] B. Danev and S. Capkun, "Transient-based identification of wireless sensor nodes," in *International Conference on Information Processing in Sensor Networks*, April 2009, pp. 25–36.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

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

[7] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?" *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, 2017.

[8] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Engineering Applications of Neural Networks*, 2016, pp. 213–226.

[9] T. J. OShea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[10] D. Roy, T. Mukherjee, M. Chatterjee, and E. Pasiliao, "Detection of Rogue RF Transmitters using Generative Adversarial Nets," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2019.

[11] T. O'Shea and N. West, "Radio Machine Learning Dataset Generation with GNU Radio," *Proceedings of the GNU Radio Conference*, vol. 1, no. 1, 2016.

[12] N. Wagle and E. Frew, "Spatio-temporal Characterization of Airborne Radio Frequency Environments," in *IEEE GLOBECOM Workshops (GC Wkshps)*, 2011, pp. 1269–1273.

[13] T. OShea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[14] J. S. Ren, Y. Hu, Y.-W. Tai, C. Wang, L. Xu, W. Sun, and Q. Yan, "Look, Listen and Learn-A Multimodal LSTM for Speaker Identification," in *AAAI*, 2016, pp. 3581–3587.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural computation*, vol. 9, no. 8, p. 17351780, November 1997.

[16] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *CoRR*, vol. abs/1412.3555, 2014.

[17] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, 2015, pp. 802–810.

[18] R. Dey and F. M. Salemt, "Gate-variants of Gated Recurrent Unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1597–1600.

[19] Ettus Research, "USRP B210," https://www.ettus.com/product/details/UB210-KIT, 2018.

[20] NooElec, "USRP B210," http://www.nooelec.com/store/sdr/sdr-receivers/nesdr-mini-rtl2832-r820t.html, 2018.

[21] T. J. O'Shea, S. Hitefield, and J. Corgan, "End-to-end Radio Traffic Sequence Recognition with Recurrent Neural Networks," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 277–281.

[22] T. J. O'Shea, T. C. Clancy, and R. W. McGwier, "Recurrent Neural Radio Anomaly Detection," *CoRR*, vol. abs/1611.00301, 2016.

[23] H. Rutagemwa, A. Ghasemi, and S. Liu, "Dynamic Spectrum Assignment for Land Mobile Radio with Deep Recurrent Neural Networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[24] S. Rajendran *et al.*, "Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 433–445, 2018.

[25] radioML, "RFML 2016," https://github.com/radioML/dataset, 2018.

[26] S. Bai, M. Yan, Y. Luo, and Q. Wan, "RFedRNN: An End-to-End Recurrent Neural Network for Radio Frequency Path Fingerprinting," in *Recent Trends and Future Technology in Applied Intelligence*, 2018, pp. 560–571.

[27] J. Toonstra and W. Kinsner, "A radio transmitter fingerprinting system ODO-1," in *Canadian Conference on Electrical and Computer Engineering*, vol. 1, 1996, pp. 60–63.

[28] D. Shaw and W. Kinsner, "Multifractal Modelling of Radio Transmitter Transients for Classification," in *IEEE WESCANEX*, 1997, pp. 306–312.

[29] I. O. Kennedy, P. Scanlon, F. J. Mullany, M. M. Buddhikot, K. E. Nolan, and T. W. Rondeau, "Radio Transmitter Fingerprinting: A Steady State Frequency Domain Approach," in *IEEE Vehicular Technology Conference*, 2008, pp. 1–5.

[30] S. Xu, L. Xu, Z. Xu, and B. Huang, "Individual Radio Transmitter Identification based on Spurious Modulation Characteristics of Signal Envelop," in *IEEE MILCOM*, 2008, pp. 1–5.

[31] GNURadio, "GNU Radio," https://www.gnuradio.org, 2018.

[32] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997.

[33] F. Chollet *et al.*, "Keras: The Python Deep Learning library," https://keras.io, 2015.

[34] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, 2016.

[35] R. Al-Rfou *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *CoRR*, 2016.

[36] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, vol. abs/1502.03167, 2015.

[37] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of International Conference on International Conference on Machine Learning*, 2010, pp. 807–814.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[39] K. He and J. Sun, "Convolutional Neural Networks at Constrained Time Cost," in *IEEE CVPR*, June 2015.

[40] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.