Semi-supervised Convolutional Neural Networks for Identifying Wi-Fi Interference Sources

Krista Longi KRISTA.LONGI@HELSINKI.FI
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki

Teemu Pulkkinen TEEMU.PULKKINEN@EKAHAU.COM
Department of Computer Science, University of Helsinki
Ekahau Oy

Arto Klami ARTO.KLAMI@CS.HELSINKI.FI
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki

Editors: Yung-Kyun Noh and Min-Ling Zhang

Abstract

We present a convolutional neural network for identifying radio frequency devices from signal data, in order to detect possible interference sources for wireless local area networks. Collecting training data for this problem is particularly challenging due to a high number of possible interfering devices, difficulty in obtaining precise timings, and the need to measure the devices in varying conditions. To overcome this challenge we focus on semi-supervised learning, aiming to minimize the need for reliable training samples while utilizing larger amounts of unsupervised labels to improve the accuracy. In particular, we propose a novel structured extension of the pseudo-label technique to take advantage of temporal continuity in the data and show that already a few seconds of training data for each device is sufficient for highly accurate recognition.

Keywords: Convolutional neural networks, Semi-supervised learning, Wi-Fi, Spectral data, Temporal data

1. Introduction

Wireless Local Area Networks (WLAN) based on IEEE 802.11 (Wi-Fi) operate in the 2.4GHz and 5GHz spectrum, and since these frequency bands are unlicensed other devices operating in the same frequency range can affect the performance of the Wi-Fi network. Identification of such interfering devices plays a crucial role both in planning the initial network configuration and in troubleshooting the network.

Since the devices cause interference in the spectral range of Wi-Fi, it is natural to attempt detecting and identifying them by monitoring the signal spectrum itself. For example, Rayanchu et al. (2011) developed the AirShark method for identifying 8 different devices by constructing decision tree classifiers that are trained individually for each possible device. While they demonstrated that it is possible to identify non-Wi-Fi devices based on generic and device-specific features that attempt to capture the spectral and temporal properties of the transmission, their approach has serious limitations in terms of practical applicability.

The first challenge is the sheer number of possible interfering devices. The United States Federal Communications Commission lists close to 500 devices that are authorized to operate in the 2.4GHz band, many of which are common in urban environments. Developing individual detectors for all of these is not feasible, but instead, we need to consider consolidated models that can detect arbitrary devices available during training of the model.

The other core challenge is in obtaining training data for learning the detector. This is typically done in laboratory conditions, by switching on a non-Wi-Fi device within proximity of the sensor for collecting training instances for recognizing that particular device. Furthermore, one should typically make several recordings for the same device in different conditions, varying e.g. the distance and medium (walls, windows, etc.) between the device and the sensor or the power of the device. This is a laborious process that scales linearly with the number of devices, and produces noisy ground truth labels for training the detector: Even if some device is switched on, it does not necessarily generate signal interference all the time. For example, microwave ovens modulate their power by switching the magnetron on or off instead of modifying the transmitted power, and transmitters and receivers may only cause interference when there is actual communication. Consequently, to obtain high-quality training data we would need to manually inspect the spectral data to identify when the device is causing possible interference.

In this work, we overcome these limitations by introducing a unified model for recognizing multiple non-Wi-Fi devices based on spectral data. We use a convolutional neural network (CNN) (LeCun et al., 1995) that does not require device-dependent feature extractors but instead learns an internal feature representation useful for classifying arbitrary devices. The model recognizes the devices from real-time streaming data based on short temporal windows of 2 seconds, can be extended to recognize new devices without re-training from scratch, and most importantly, can be trained based on very limited supervision labels. Instead of requiring access to fully labeled training collection, our model operates in a semi-supervised fashion: For each device, we only require a few isolated examples of 2 second windows when it was known to be broadcasting a signal, together with unlabeled data recorded during times when some device was likely on but the labeling cannot be verified without manual evaluation. To collect this kind of data it is sufficient to merely leave the devices on for a longer time and evaluate the true labels for a few isolated spectral windows.

We apply CNNs for the problem by sliding a window over the incoming signal. For any given time point, the input for the model is a two-dimensional image where one dimension is over the frequencies and the other over time within the window. CNNs have been used for spectral data in a similar fashion in speech recognition (Abdel-Hamid et al., 2014), but to our knowledge not for spectral data recorded for radio frequencies. The core technical contribution of the work is a new semi-supervised learning technique designed for temporal data. We build on the concept of *pseudo-label* by Lee (2013), which allocates a pseudo-label for each training instance without a known label and alternates between updating the pseudo-labels based on the current classifier and updating the classifier based on both the true training labels and the pseudo-labels. The standard pseudo-label algorithm assigns the pseudo-labels independently for each instance. We extend the concept into *structured pseudo-labels* that take into account relationships between the training instances. In particular, we develop an extension that supports temporal continuity in the pseudo-labels by assuming they satisfy a

Markovian assumption. This extension is shown to increase the final classification accuracy when operating with a very small number of reliable training samples.

In the following, we first explain related work regarding non-Wi-Fi device detection and semi-supervised learning and then proceed to explain our solution for the problem. We evaluate the model on real spectral data in Section 4 and show that we are able to reach essentially perfect recognition for 13 different devices with a unified model trained only with tens of seconds of labeled data for each device. We outperform a simplified version of the work by Rayanchu et al. (2011) on our data already with just a few seconds of training data for each device, and demonstrate that structured pseudo-labels help in reducing the error by one-third when operating with few labeled instances.

2. Background

This section presents relevant background for our work on identifying non-Wi-Fi radio frequency devices from spectral data. In Section 2.1 we present related work in interference detection. Our solution to the problem is a unified convolutional neural network that can be trained even with limited amounts of labeled data, and consequently, we briefly describe CNNs for spectral data in Section 2.2 and discuss methods for coping with small amounts of training data in Section 2.3.

2.1. Interference Detection

Though Wi-Fi devices are able to operate in both 2.4GHz and 5GHz bands, the focus on interference detection has traditionally been on the 2.4GHz band. This frequency band is more crowded since it was unlicensed earlier and has more directly competing technologies (e.g. Bluetooth, Zigbee). The quality of a Wi-Fi link can be affected by the hundreds of devices operating in the spectrum (The United States Federal Communications Commission), and these devices are common in many different environments (Rayanchu et al., 2011). For instance, household microwave ovens use this frequency specifically because it is unlicensed. Frequency-hopping and spread-spectrum techniques are also more frequent in this band.

Devices implementing the IEEE 802.11 specification have mechanisms in place for coping with concurrent transmissions; a clear-channel mechanism detects within-specification transmission on the same channel and can withhold communication based on timing values described in the received transmissions, and an energy-detection mechanism can be used to measure non-Wi-Fi energy. Use of these techniques is considered standard operation in the network and they do not affect the overall data rate. Strong interference, especially outside of the Wi-Fi specification, however, can corrupt packet preambles or cause frame error checks to fail, necessitating retransmissions. The standard response is for wireless stations to reduce the complexity of modulation used for transmission (dynamic rate switching), which has the effect of lowering the bit rate of the communication. This can have a cascading effect since transmitting at a lower rate will increase the time during which the wireless channel is occupied. This in turn impacts even devices which are not directly sensing interference. Constant strong interference will thus lower the overall throughput of the network.

Identification of such interfering devices is important for both planning the initial network configuration, such as the placement of the network stations, as well as in troubleshooting the network in case of poor performance. Often the cause is an individual non-Wi-Fi radio frequency device somewhere within the network, and the network can be repaired by isolating or removing the device, lowering its transmission power or reconfiguring the channels in use to avoid the source of interference. Previous works have sought to quantify interference (Rayanchu et al.), to identify the devices causing interference (Rayanchu et al., 2011; Kim et al., 2014) and to locate the interference source (Rayanchu et al.).

The closest to our work is the AirShark system (Rayanchu et al., 2011), which can recognize eight different non-Wi-Fi devices operating in the 2.4GHz band based on a set of generic features (frequency, bandwidth, spectral signature, duty cycle, pulse signature, inter-pulse timing signature, pulse and spread) as well as some device-specific features like sweep detection. By training individual decision trees and further tuning the features for each device, the method is able to reach 91–100% recognition accuracy depending on the device. While this is sufficient for practical applications, the approach is limited by the need to select the features and train a separate classifier for each device. In Section 4, we will compare our approach against a baseline motivated by AirShark on our data, demonstrating that we outperform it by a wide margin. Our system scans 100MHz band with a resolution bandwidth of 40kHz once per 40ms, whereas AirShark was initially proposed for a device scanning 20MHz channels with a 312.5kHz resolution once per 100μs. Hence we cannot incorporate all of AirShark's features related to short time-scales, but our implementation benefits from the wider frequency band compared to the original algorithm.

There are also other systems using specialized hardware to detect interfering devices (Hong and Katti, 2011; Gollakota et al., 2011). A different approach for troubleshooting Wi-Fi network performance and identifying the type of device causing interference is to combine packet loss analysis and 802.11 ACK number analysis (Kim et al., 2014). Cyclostationary aspects of the signal have also been used for detection. Autocorrelation of peaks in the spectral data was used to detect microwave ovens by looking for periodicities matching the AC oscillating frequency of 60Hz (Weng et al., 2014). Commercial products include AirMagnet's Spectrum XT and Cisco's CleanAir Technology, which can coordinate detection and localization between compatible wireless access points.

2.2. Convolutional Networks for Spectral Data

The convolutional neural networks are naturally suited for analysis of data where spatial or temporal invariance of objects within a larger structure is needed, such as image classification. Quite naturally, they have been successfully applied for spectral data as well, in particular in audio and speech recognition (Abdel-Hamid et al., 2014). The audio data is converted into a spectrogram by computing short-time Fourier transforms, and the resulting spectrogram is analyzed by interpreting temporal time windows as images that range over frequencies and time. We use the same approach for analyzing the spectral data in our application, but here the frequency range is provided by the measuring device instead of computing the Fourier transform from an audio waveform.

The convolutions in such a model can be computed either over the frequency domain or the temporal domain, or both. For example, Abdel-Hamid et al. (2012) and Abdel-Hamid et al. (2014) considered convolutions only over the frequency dimension to normalize speaker variance in speech recognition while modeling the temporal continuity with hidden Markov models (HMMs), whereas Lee et al. (2009) computed convolutions over the temporal domain to learn phonemes from unlabeled data. The choice ultimately depends on the nature of the application; we use two-dimensional convolutions over both the spectral and temporal dimension to learn features that are invariant of the specific frequency of the interfering device and placement of the analysis windows.

2.3. Coping with Small Data

The success of CNNs has traditionally been based on large amounts of labeled training data (Krizhevsky et al., 2012). However, collecting such data can be expensive or impractical, and methods for increasing the data or utilizing unlabeled data are often required. There are three main approaches for this: pre-training with unlabeled data (Erhan et al., 2010; Lee et al., 2009), augmenting the labeled data with some transformations (Krizhevsky et al., 2012) and utilizing both labeled and unlabeled data in the training (Chapelle et al., 2006). We shortly describe the first two methods and explain why they are not promising candidates in the application of Wi-Fi interference identification, and then proceed to explain in more detail the techniques proposed for actual semi-supervised learning that use both supervised and unsupervised samples.

The key idea in unsupervised pre-training is to use a large amount of unlabeled data to learn a good feature representation, which then helps to solve the supervised learning problem on the more limited labeled set (Erhan et al., 2010). However, unsupervised pre-training is in itself a harder task than directly training a supervised model, often requiring a generative model for the data. It is hence feasible for domains where a) good generative models exist, and b) a number of unsupervised instances is truly massive to facilitate solving a more complex learning task. In our application this is not the case; even unsupervised training data needs to be collected largely manually, and no good generative formulations for arbitrary Wi-Fi spectrum data exist. Similarly, augmenting the training data by applying various transformations for the limited training samples, shown to improve accuracy for example in image classification (Krizhevsky et al., 2012), is not applicable to our case. For natural images it is easy to imagine transformations that preserve the labels, for example by modifying the contrast or by adding camera noise, but for spectral data we cannot tell which kinds of rich enough transformations would be guaranteed to preserve the labels.

We hence turn to standard semi-supervised learning algorithms that use both labeled and unlabeled data during the training phase (Chapelle et al., 2006). A practical family of semi-supervised algorithms is based on alternatively inferring the labels for the unlabeled samples based on the current model and updating the model itself based on both the actual supervised labels and the labels inferred for the unlabeled samples; for early works see Ghahramani and Jordan (1994); Blum and Mitchell (1998). Our solution builds on *pseudo-labels* proposed by Lee (2013) as a simple semi-supervised strategy for deep learning. They trained the model with sure labels, and in each iteration added pseudo-labels for unlabeled data by choosing the class with maximum predicted probability for each data point. These pseudo-labels were then used in training as if they were real labels, with additional weighting for the importance of the individual samples based on the amount of the unlabeled data and the phase of learning. The technique has been used, for example, to improve image classification accuracy on small datasets by augmenting the data by images queried from online image search (Kolář et al., 2016).

More advanced semi-supervised learning techniques have been proposed recently, typically demonstrating improved accuracy on classical benchmark datasets. The results achieved by Lee (2013) were further improved with pseudo-ensemble agreement (PEA) regularization (Bachman et al., 2014) that was able to reduce the test error from the 10.49% achieved by pseudo-labels to 5.21% on the MNIST data with 100 labeled samples, and Rasmus et al. (2015) used semi-supervised learning with Ladder Networks to reach test error of 1.06%. In this work, we build on the pseudo-label technique due to its' simplicity and expandability, and because it is sufficient for solving our application with essentially perfect accuracy.

3. Methods

Building on the background presented in the previous section, we now describe the proposed solution for identifying the interference sources. We first describe the nature of the input data so that the design decisions for the proposed model can be understood, and then proceed to describe the convolutional neural network used for classifying the samples, followed by the presentation of the semi-supervised learning approach and the new structured extension for the pseudo-label idea.

3.1. Input

The data was collected with a spectrum analyzer provided by Ekahau. The spectrum analyzer was configured to perform sweeps over the 2.4GHz spectrum at a resolution bandwidth (RBW) of about 40kHz, which it provided at a rate of about 25Hz, or once per 40ms. The 2.4GHz band (spanning 100MHz) thus provides roughly 3000 individual features, each corresponding to a bin in fast Fourier transform, for every time slice. Such individual time slices are not alone sufficient for recognizing the devices since some devices are only recognized based on their temporal characteristics, and hence we analyse the data in form of two-second windows, each collecting 50 consecutive slices. For learning we use partially overlapping windows, starting a new one every second. For an illustration of the raw spectral data, see Figure 1 (left). The only preprocessing done for the raw data is z-score normalization based on the no-interference samples.

The data consists of measurements of 13 different devices. The devices included in the data are a wireless video camera, a microphone, a remote control for a remote-controlled car, a headset and an intercom, two baby monitors, a microwave oven, two jammers and three wireless spy cameras. For each device, we collected 4–6 different measurements varying the distances or other conditions. Each measurement consists of roughly three minutes of data: one-minute recording of the device turned on surrounded by one minute periods of background recording with no active interfering devices. For cases where the device setup consisted of a transmitter and a receiver, the transmitter was turned on before the receiver.

A critical property of this data collection procedure, shared by all reasonable ways of collecting training data for solving the problem, is that the actual onset and offset times for interference are not known precisely. The devices are turned on manually, with no time synchronization with the measurement device, and the actual interference may start at a delay depending on the device or be sporadic as in the case of microwave oven operating at low power. Consequently, we cannot directly use the recorded onset and offset times for providing the training labels, and in fact, we did not even attempt to carefully record them.

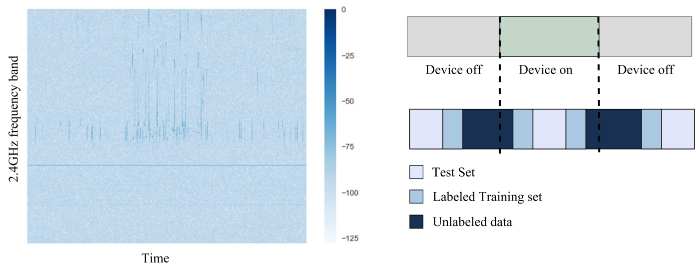

Figure 1: Example of one data measurement of microwave oven at 800w (left) and the division of one measurement into test and training sets (right).

Instead, we conducted the experiments in a way where only data with guaranteed labels are used for testing, whereas the training is done on portions of the data for which the true labels are uncertain, letting the semi-supervised learning technique infer the real labels.

For each measurement of three minutes, we extracted one minute of test samples by taking the first and last 20 second periods to represent no interference and the middle 20 second period to represent the device causing interference. Despite our inaccurate onset and offset timings, these samples are guaranteed to match the correct class. The remaining samples were left for training. For the training instances, we set the label according to the imprecise onset/offset timing of the device. Since the labels for samples near the onsets and offsets are uncertain, we construct several versions of the training set by controlling the number of labeled samples. Figure 1 (right) demonstrates the division of one measurement into these sets, clarifying the way the training samples are assigned into labeled and unlabeled sets.

In summary, our data collection has for most devices 40 non-overlapping or 76 overlapping windows per measurement used for testing, combined with 80 or 152 windows of no-interference. The training set consists of roughly double the amount of windows per measurement, of which roughly half can be reliably assigned with true labels. In the experiments we evaluate the model also with a considerably smaller number of labeled examples, to estimate how little data is sufficient for learning the model.

3.2. Convolutional Neural Network

As described above, the input for the classifier is provided as two-second windows each corresponding to 50 consecutive time-slices that are each a 3000-dimensional vector of real-valued numbers. Such a window can be interpreted as a 50x3000 image, allowing us to borrow network design principles from standard applications of convolutional neural networks for image data. Since the total amount of data in our experiments is fairly limited for a deep learning model, only about 10000 instances in total including the unlabeled instances, we use a relatively shallow network while still allowing for clearly non-linear decision surfaces and sufficient structure for learning reasonable latent representation. The convolutions are used for modeling features that are broader than one frequency bin or time-slice, and the standard choice of max pooling is used to implement temporal invariance within the window.

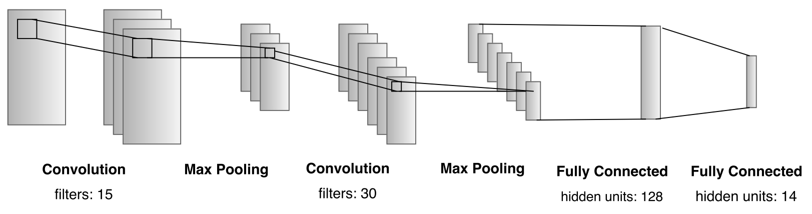

Figure 2: Structure of the convolutional neural network. In all layers, the filter size used was 3x3 and stride 3.

The architecture used in the empirical experiments is presented in Figure 2 and consists of two convolutional layers with rectified linear unit activation, two max-pooling layers, and a fully connected layer. Finally, the output from the last layer is given to a 14-way softmax that gives a distribution over the labels. We also use drop-out as well as ℓ_2 regularization in the convolutional layers to avoid overfitting. For training, we used stochastic gradient descent with a mini-batch size of 50 and a learning rate 0.1. In a set of preliminary experiments, we found that a wide range of alternative structures would solve the problem well, indicating that the solution is robust for minor modifications. Very shallow networks, however, were not sufficient for solving the classification problem, and leaving out the convolutions dramatically reduced the performance. With larger datasets, we could conceivably user deeper networks.

3.3. Semi-Supervised Learning with Pseudo-Labels

As described above, our data comes in form of a sequence of time windows where the reliability of the supervising label depends on the time; for instances around the device onset or offset point we cannot tell the true label, whereas for the ones far away from those we can reliably say that the device was on. Note, however, that even this does not guarantee that the label is necessarily correct since it may not be causing interference at that time.

To cope with such data, we use pseudo-labels so that the samples with sufficiently certain labels are provided as supervised examples, whereas the rest are left as unsupervised samples for which we learn pseudo-labels during training. We first explain the standard pseudo-label technique by Lee (2013), and then proceed to present our extension for structured domains, here samples with temporal continuity.

3.3.1. Pseudo-Labels

Pseudo-Labels (Lee, 2013) is a simple method for utilizing unlabeled data in training deep neural networks. It is an EM-like solution where in each iteration the class with the maximum predicted probability is given as the label for the unlabeled samples, which are then used in the training in a supervised fashion. The full learning algorithm alternates between solving the supervised learning problem using all samples and assigning new pseudo-labels for the unsupervised samples.

Algorithm 1 Structured Pseudo-Label

```
Data: labeled data $x$, labels $y$, unlabeld data $x'$
Function trainModel($x, y, x'$)
    model ← train using $x, y$
    $y'$ ← predict(model, $x'$)
    while $y'$ changes do
        model ← train using $x + x', y + y'$
        $y'$ ← predict(model, $x'$)
    end
Function predict(model, x)
    $\hat{y}$ ← predict labels for $x$
    $y$ ← viterbi($\hat{y}$)
    return $y$
```

The whole optimization problem can be defined using a consolidated loss function

$$\mathcal{L}(\theta, y') = \frac{1}{n}\sum_{m=1}^{n} L(y_m, f_m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'} L(y_m^{'}, f_m^{'}), \tag{1}$$

where n is the number of samples in labeled data and n' in unlabeled data, f_m is the outputs and y_m the true labels of labeled data, and $f_m^{'}$ is the outputs and $y_m^{'}$ the pseudo-labels of unlabeled data.

The first term in the loss corresponds to a classical supervised loss, whereas the latter term that is optimized over both the pseudo-labels y' and the network parameters implements the semi-supervised learning. The coefficient $\alpha(t)$ is used to scale the weight given for pseudo-labels in the loss function. Lee (2013) increase the value of $\alpha(t)$ slowly during the process, however, in our case, just using a constant value $\alpha(t) = 1$ was sufficient.

3.3.2. Structured Pseudo-Labels

Our core technical contribution is an extension of pseudo-labels that takes into account the structure underlying the sample space. Instead of independently assigning the pseudo-labels for each sample, we introduce in (1) an additional term that encourages solutions that are smooth over the sample space and then assign the pseudo-labels for all samples at once. In this work we consider temporally continuous samples and present the technical details for labels that exhibit a Markovian property but note that the same principle applies also, for example, spatial continuity implemented using a Markov random field.

The temporal continuity is enforced by assuming a Markovian property: The distribution of possible labels depends on the label of the previous sample. This is incorporated into the loss function (1) by adding a new term

$$\lambda \sum_{m=2}^{n'} \mathbb{I}[y_i^{'m} \neq y_i^{'m-1}], \tag{2}$$

which penalizes the model by λ for every case where consecutive samples are assigned a different label. Here $\mathbb{I}$ is the identity function that evaluates to 1 when the condition holds and to 0 otherwise.

Given this term, the pseudo-labels can no longer be assigned independently. However, a straightforward dynamic programming algorithm corresponding to the Viterbi algorithm used for Hidden Markov models (HMMs) can be used for making the assignment. This is natural given that the structured pseudo-label loss corresponds to the temporal structure of a simplified HMM where all self-transitions share the same probability p and the remaining transitions are all set to equal value $q = (1-p)/C$, where C is the number of devices[1]. The prior negative log-probability of a sequence under such a transition matrix is given by

$$\begin{aligned}
&-\sum_{m=2}^{n} \mathbb{I}[y_i^{'m} \neq y_i^{'m-1}] \log q - \sum_{m=2}^{n} \mathbb{I}[y_i^{'m} = y_i^{'m-1}] \log p \\
&= -\sum_{m=2}^{n} \mathbb{I}[y_i^{'m} \neq y_i^{'m-1}](\log q - \log p) - \sum_{m=2}^{n} \log p \\
&= \sum_{m=2}^{n} \mathbb{I}[y_i^{'m} \neq y_i^{'m-1}](\log p - \log q) + D,
\end{aligned}$$

where the constant D does not depend on the assignments. Hence, λ in (2) relates to the transition probabilities by $\lambda = \log p - \log q$, where $p > q$ to indicate preference for retaining the same label. The larger λ is, the bigger the preference for retaining the label. In our experiments we set λ to 1.18, corresponding to $p = 0.2$, but as demonstrated in Section 4 the technique is very robust for the choice. Note that $p = q$ results in $\lambda = 0$, reverting back to the classical pseudo-label technique.

The structured pseudo-label algorithm (Algorithm 1) hence optimizes for the loss that combines (1) and (2), by alternating between learning of the network parameters given both the true labels and the pseudo-labels and inferring the pseudo-labels with the Viterbi algorithm. Finally, the same Viterbi algorithm can be used for encouraging temporal continuity of the labels also in the test set.

4. Experiments

In the following, we conduct a series of experiments to demonstrate the proposed algorithm. We first present a simplified version of AirShark (Rayanchu et al., 2011) to act as a baseline, and then proceed to show that the proposed convolutional neural network solves the core classification problem with near perfect accuracy whereas the baseline reaches only 80% accuracy when both methods are allowed to use a maximally large number of labeled samples. To further highlight the advantage of our consolidated model (instead of device-specific models of AirShark), we demonstrate that the latent representation learned by the convolutional network generalizes over devices, by showing that when adding new devices it is sufficient to re-train only the last layer of the network.

We then proceed to study the effectiveness of the pseudo-label technique and its newly proposed structured variant in coping with limited training samples. We experiment with varying sizes of labeled training instances and show that pseudo-labels clearly help with small training sets.

1. Note that the number of classes in the classification problem is C+1, since we also have the no-interference scenario.

CNN Classifier

True Label		off	video	mic	head	inter	baby1	baby2	mwo	bbjam	nbjam	spy1	spy2	spy3	rc
	off	1077	0	1	0	0	0	0	2	0	0	0	0	0	0
	video	5	35	0	0	0	0	0	0	0	0	0	0	0	0
	mic	0	0	39	1	0	0	0	0	0	0	0	0	0	0
	head	0	0	6	34	0	0	0	0	0	0	0	0	0	0
	inter	9	0	0	0	31	0	0	0	0	0	0	0	0	0
	baby1	0	0	0	0	0	40	0	0	0	0	0	0	0	0
	baby2	0	0	0	0	0	0	40	0	0	0	0	0	0	0
	mwo	14	0	0	0	0	0	0	26	0	0	0	0	0	0
	bbjam	0	0	0	0	0	0	0	0	40	0	0	0	0	0
	nbjam	0	0	0	0	0	0	0	0	0	60	0	0	0	0
	spy1	0	0	0	0	0	0	0	0	0	0	40	0	0	0
	spy2	0	0	0	0	0	0	0	0	0	0	0	40	0	0
	spy3	0	0	0	0	0	0	0	0	0	0	0	0	40	0
	rc	6	0	0	0	0	0	0	0	0	0	0	0	0	34

Predicted Label

Baseline Classifier

True Label		off	video	mic	head	inter	baby1	baby2	mwo	bbjam	nbjam	spy1	spy2	spy3	rc
	off	778	1	15	47	0	0	1	43	0	0	0	0	0	195
	video	0	30	0	0	0	0	0	0	0	0	0	0	0	10
	mic	0	0	40	0	0	0	0	0	0	0	0	0	0	0
	head	0	0	4	36	0	0	0	0	0	0	0	0	0	0
	inter	8	0	0	3	29	0	0	0	0	0	0	0	0	0
	baby1	0	0	0	0	0	40	0	0	0	0	0	0	0	0
	baby2	0	0	0	1	0	1	38	0	0	0	0	0	0	0
	mwo	0	0	0	0	0	0	0	27	0	0	0	0	0	13
	bbjam	0	0	0	0	0	0	0	0	40	0	0	0	0	0
	nbjam	1	0	0	0	0	0	0	0	0	59	0	0	0	0
	spy1	0	0	0	0	0	0	0	0	0	0	40	0	0	0
	spy2	0	0	0	0	0	0	0	0	0	0	0	40	0	0
	spy3	0	0	0	0	0	0	0	0	0	0	0	0	40	0
	rc	0	0	0	0	0	0	0	2	0	0	0	0	0	38

Predicted Label

Table 1: The confusion matrix of a basic CNN classifier (left) and a baseline classifier (right). The classification accuracy is 97% with CNN and 79% with the baseline.

4.1. Baseline

We present a baseline classifier based on the current state of the art method AirShark (Rayanchu et al., 2011), modified to match the somewhat different raw data recorded by the Ekahau device. Specifically, we use the "spectral signature" feature to describe each device's spectrum use and their angular difference as a similarity metric. AirShark describes 8 features in total, but some of them are based on aspects of the sampling unavailable to us (e.g. <1ms "inter-pulse time" for Bluetooth, 16.66ms sweeping pattern for microwave ovens).

The split into training and testing datasets was performed as detailed in section 3.1, except that no overlap between the samples was used. In the training phase, the average spectral sample was calculated over the section of data where we could be sure the device was turned on. A separate "off" signature was created by merging all spectral samples where devices were known to be off. This signature could be seen to represent the background/device noise detected during measurements. Then, the spectral signature $\hat{s}$ was calculated as

$$\hat{s} = \frac{\boldsymbol{s}}{\|s\|},$$

where $\boldsymbol{s}$ is a vector representing the power in each bin, or in other words, the spectral sample in the form of a unit vector.

Detection was performed by calculating the angular difference between all training signatures and the spectral signature of the testing windows and picking the signature with the smallest angle. The results are presented in Table 1. The baseline classifier has issues in particular with devices that are not constantly transmitting (RC, MWO) or are frequency hopping (MIC, HEAD). Though the devices are turned on, the spectral samples might not contain signal from the device, or might not stand out distinctly enough within the testing window. This would explain the confusion with respect to background samples ("off").

4.2. Experiment 1: Interference Classification on Sufficiently Large Data

To demonstrate the performance of the proposed CNN architecture we compare it against the baseline described above. We use a maximally large number of training instances to

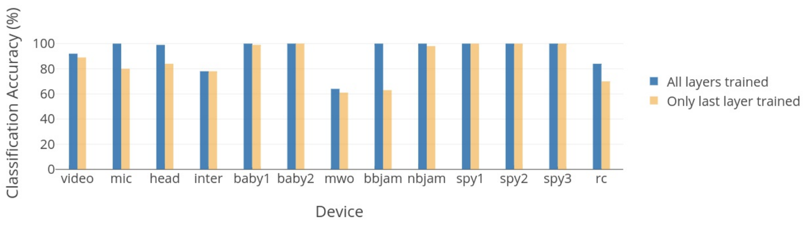

Figure 3: Classification accuracy on the multiclass classifier for each device when the training data from that device was included in training the whole network (blue) or just used to train the outer layer (orange).

show how the basic classification model operates when the number of training labels is not critically low. The proposed model achieves 2% error rate compared to 21% error for the baseline, indicating a clear qualitative difference. Analysis (see Table 1) reveals that the baseline is accurate for some devices, but difficulties with a few devices lower the overall accuracy dramatically. Furthermore, we see that the majority of the mistakes for the proposed method is in classifying some microwave oven samples as "off". This is explained by the way microwave ovens modulate their power by switching the magnetron off, and hence the predictions can be considered to be correct and the ground truth labels wrong.

To further illustrate the proposed CNN, we run an additional experiment where we train the model only on a subset of devices, and then extend it to new devices by only re-training the last layer of the network to classify a new device not available during training. To do well in this task the convolutional layers must have learned rich internal representation capturing essential properties of interfering devices. It also paves way towards practical maintainable detection tool, as needing to re-train the whole model from scratch when learning to recognize a new type of a device saves computation time. For each device, we trained the network with data from all the other devices and then used the training data from that device to retrain only the outer layer. The results are presented in Figure 3. With most devices, we reach equal or nearly equal classification accuracies to when the whole network was trained with that data, indicating that the above claims are true for the proposed architecture.

4.3. Experiment 2: Semi-supervised Learning

As there are hundreds of possible interfering devices, and collecting accurately labeled training data from all of those devices is expensive and impractical, we run experiments to see how much data is required for learning sufficiently accurate models and to evaluate the importance of using unlabeled data points while training the model. We trained the model with varying sizes of labeled training data where the device was turned on, starting with 4 seconds per measurement up to 72 seconds per measurement, corresponding to roughly all of the data recorded when the device was supposedly on according to the imprecise timings used during the data collection. We then compared three alternative methods: Pure supervised learning that ignores the unlabeled samples, the classical pseudo-label technique by Lee (2013), and the structured pseudo-label technique described in Section 3.3.2.

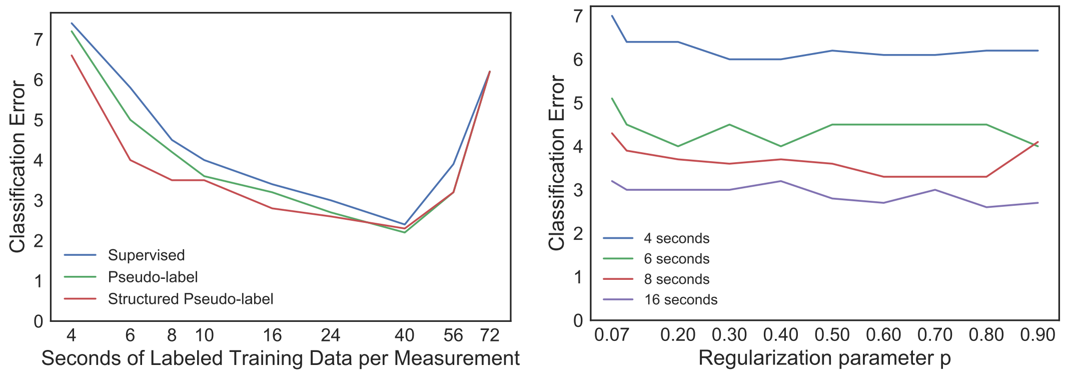

Figure 4: (**Left:**) Classification error percentage with different amounts of labeled training data. With small amounts of labeled data (up to about 20 seconds per measurement), structured pseudo-labels with $p = 0.2$ (red curve) reduce the test classification error compared to pseudo-labels or just using labeled data. With larger amounts of data, all methods perform equally. (**Right:**) Classification error using structured pseudo-labels with different values for the regularization parameter p and different amounts of labeled data. The value $p \approx 0.07$, or equivalently $\lambda = 0$, corresponds to regular pseudo-labels and the structured case outperforms it for all values of p between 0.1 and 0.9.

Figure 4 (left) illustrates the error rates with different amounts of training data. When using less than 10 seconds of training data for each measurement, using unlabeled samples clearly helps and imposing the temporal continuity in labels using the structural version further improves the accuracy. For the case of 6 seconds of data per measurement, the purely supervised version has an error of 5.8%, which is reduced to 5.0% by using pseudo-labels and to 4.0% with the structured variant. Also note that when using more than 40 seconds of training data per measurement the accuracy declines – this is because some of the training labels assumed to be certain are already wrong. The pseudo-label technique does not help here because it does not modify the training labels, but it could be extended to mitigate this issue by allowing refinement of the labeled examples as well. Figure 4 (right) presents the classification error as a function of the regularization parameter p for various setups, showing that the method is very robust for the choice. The structured variant outperforms standard pseudo-label ($\lambda = 0$ or $p = q = 1/14$) for any choice of $p > q$.

To further illustrate the importance of modeling the temporal continuity, we present in Figure 5 a simulated example of applying the model in sequentially appearing data. We concatenated randomly chosen measurements into a sequence of continuous measurement, trained the model using 8 seconds of labeled data for each measurement not included in the sequence, and visualize the predictions in the real temporal order. Imposing the temporal constraint both during training and testing clearly reduces isolated misclassified samples during the time a device is on.

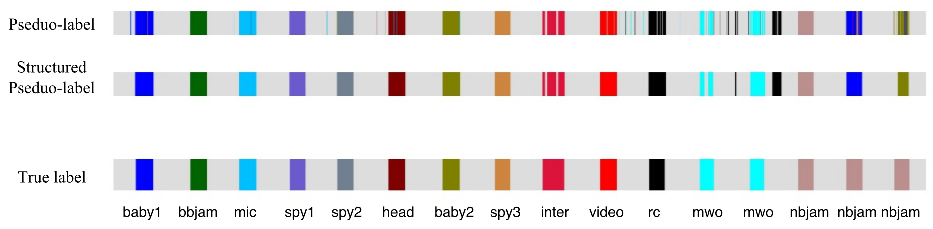

Figure 5: Performance with pseudo-labels and structured pseudo-labels on a realistic situation. Imposing the temporal constraint both during training and testing clearly reduces isolated misclassified samples.

5. Discussion

In this paper, we have presented a solution for identifying non-Wi-Fi radio frequency devices from spectral data. Wi-Fi operates in an unlicensed spectrum, and thus various devices that are common in urban environments can affect the performance of the network. Identification of such interfering devices is important for planning the network configuration as well as for troubleshooting. There are several challenges related to this. First, there are hundreds of such devices and thus the classification method needs to be easily adapted for new devices. In addition, collecting enough data for the training from all these devices is impractical, and the most natural ways of collecting training data provide ground truth labels for only a subset of the samples. Therefore, having an algorithm that can be easily trained with small amounts of labeled data while still utilizing also unlabeled samples is important.

To address these challenges we proposed a convolutional neural network for recognizing non-Wi-Fi devices, and in particular, proposed a novel semi-supervised learning technique for learning the model from very scarce training labels. The new technique extends pseudo-labels (Lee, 2013) for structured sample domains; we presented here the details for outputs that are smooth in time, but the concept can be generalized for more complex dependencies.

The proposed model was shown to achieve near perfect accuracy in recognizing 13 different devices, even when trained with only a few seconds of labeled data for each device. With slightly more data the model reaches effectively perfect accuracy, and it does so by learning a device-independent latent representation that can be generalized for new devices by merely re-training the last layer of the network. It hence effectively mitigates the main challenges in training models for detecting wireless interference and acts as a reasonable starting point for practical production solutions.

Perhaps the core limitation of this work is that because we only have access to training data with only one device on at a time, we could not evaluate the performance of the model in setups where more than one device can cause interference. In troubleshooting, which is the primary use-case for the recording device, detecting isolated interference is typically sufficient, since it is unlikely that several new devices would start operating within the network at the same time. For planning purposes a model that can recognize multiple overlapping interference signals would be useful. However, interference is typically localized because of the inverse square law for signal attenuation, and hence multiple interfering devices would have to be co-located to cause simultaneous interference. The proposed solution generalizes directly for setups with multiple devices on simultaneously, requiring only minor changes in the overall loss function and the dynamic programming algorithm for assigning the pseudo-labels, but training and evaluating the model requires collecting more data that covers combinations of multiple overlapping devices.

Another limitation is that in this work we considered only the problem of detecting the presence of a device. For improved troubleshooting, it would be valuable to know also the degree of interference. The model itself is easy to extend to estimate also the signal strength, the distance to the device, and other related aspects useful for the troubleshooting activity, but again the core challenge is in collecting the training data; we leave this for future work.

Acknowledgments

We acknowledge support from Academy of Finland (grants 251170 and 266969) and Tekes (Scalable Probabilistic Analytics), as well as the computational resources provided by the Aalto Science-IT project.

References

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pages 3365–3373, 2014.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. Semi-supervised learning. 2006.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Zoubin Ghahramani and Michael I Jordan. Supervised learning from incomplete data via an EM approach. In *Advances in neural information processing systems*, pages 120–127, 1994.

Shyamnath Gollakota, Fadel Adib, Dina Katabi, and Srinivasan Seshan. Clearing the rf smog: making 802.11 robust to cross-technology interference. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 170–181. ACM, 2011.

Steven Siying Hong and Sachin Rajsekhar Katti. Dof: a local wireless information plane. In *SIGCOMM Computer Communication Review*, volume 41, pages 230–241. ACM, 2011.

Kyung-Hwa Kim, Hyunwoo Nam, and Henning Schulzrinne. Wislow: A wi-fi network performance troubleshooting tool for end users. In *INFOCOM*, pages 862–870. IEEE, 2014.

Martin Kolář, Michal Hradiš, and Pavel Zemčík. Deep learning on small datasets using online image search. In *Proceedings of the 32nd Spring Conference on Computer Graphics*, pages 87–93. ACM, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

Shravan Rayanchu, Ashish Patro, and Suman Banerjee. Catching whales and minnows using wifinet: deconstructing non-wifi interference using wifi hardware. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*.

Shravan Rayanchu, Ashish Patro, and Suman Banerjee. Airshark: detecting non-WiFi RF devices using commodity WiFi hardware. In *ACM SIGCOMM conference on Internet measurement*, pages 137–154. ACM, 2011.

The United States Federal Communications Commission. Searchable FCC ID Database. https://fccid.io/. Accessed: 2017-04-19.

Zhiyuan Weng, Philip V. Orlik, and Kyeong Jin Kim. Classification of wireless interference on 2.4GHz spectrum. In *IEEE Wireless Communications and Networking Conference*, pages 786–791. IEEE, 2014.