University of California
Santa Barbara

Towards Practical Data-Driven Network Design

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Zhijing Li

September 2019

The Dissertation of Zhijing Li is approved.

Professor Xifeng Yan

Professor Ben Y. Zhao, Committee Co-Chair

Professor Haitao Zheng, Committee Co-Chair

June 2019

Towards Practical Data-Driven Network Design

To my family, friends, and loved ones

Acknowledgements

First, I would like thank my advisors Prof. Ben Y. Zhao and Prof. Heather Zheng for their guidance, encouragement and support throughout my PhD years. Not only did they teach me to become an independent researcher, in which they trained me to focus on the critical aspects of the research projects and formulate my thoughts in the right way, but also they themselves have been role models for me in life. I have learnt innumerable lessons and gained great experience both academically and in life, which are valuable in both my career and life.

Second, I am grateful to my committee member, Prof. Xifeng Yan, for providing valuable feedback and advise on my MAE (Major Area Exam), defense and future career.

Third, it was my greatest pleasure working with my great collaborators and internship mentors. I would like to thank Jia Wang, Zihui Ge, Ajay Mahimkar, Joanne Emmons, and Laura Ogden from AT&T Research Labs, Prof. Xia Zhou from Dartmouth University, and Prof. Pedro Lopez from University of Chicago. It was amazing experiences to work on the projects together, and I learnt a lot from these outstanding researchers. This dissertation would not be possible without their earnest support.

I was very fortunate to work with a great group of colleagues in SAND Lab. I would like to thank Ana Nika, Yibo Zhu, Bolun Wang, Yanzi Zhu, Xinyi Zhang, Yuanshun Yao, Zhujun Xiao, Yuxin Chen, and Huiying Li for working together with me.

I would also like to thank all members of SAND Lab, apart from names I mentioned before, I would like to thank Xiaohan Zhao, Zengbin Zhang, Gang Wang, Qingyun Liu, Shiliang Tang, Jenna Cryan, Emily Wilson, and Max Liu. I really enjoyed all the joy and fun that they bring to me.

Finally, I would like to thank my family, my friends, and my loved ones for their understanding and unreserved support throughout my PhD years.

Curriculum Vitæ
Zhijing Li

Education

2019	Ph.D. in Computer Science, University of California, Santa Barbara.
2010	Bachelor of Science in Computer Science, Shanghai Jiao Tong University, China.

Experience

07/2015 - 06/2019	Research Assistant, University of California, Santa Barbara.
09/2014 - 06/2014	Teaching Assistant, University of California, Santa Barbara.
06/2018 - 09/2018	Research Intern, Facebook Inc.
06/2017 - 09/2017	Research Intern, AT&T Lab - Research.

Publication

“Scaling Deep Learning Models for Spectrum Anomaly Detection”. **Zhijing Li**, Zhujun Xiao, Bolun Wang, Ben Y. Zhao and Haitao Zheng, Proceedings of 20th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Catania, Italy, Jul 2019 .

“Predictive Analysis in Network Function Virtualization”. **Zhijing Li**, Zihui Ge, Ajay Mahimkar, Jia Wang, Ben Y. Zhao, Haitao Zheng, Joanne Emmons and Laura Ogden, Proceedings of 18th ACM SIGCOMM Internet Measurement Conference (IMC), Boston, USA, Nov, 2018.

“Adversarial Localization against Wireless Cameras”. **Zhijing Li**, Zhujun Xiao, Yanzi Zhu, Irene Pattarachanyakul, Ben Y. Zhao and Haitao Zheng, ACM Workshop on Mobile Computing Systems and Applications (HotMobile), Arizona, USA, February, 2018.

“Identifying Value in Crowdsourced Wireless Signal Measurements”. **Zhijing Li**, Ana Nika, Xinyi Zhang, Yanzi Zhu, Yuanshun Yao, Ben Y. Zhao and Haitao Zheng, Proceedings of the 15th International World Wide Web Conference (WWW), Perth, Australia, April 2017.

“Empirical Validation of Commodity Spectrum Monitoring”. Ana Nika, **Zhijing Li**, Yanzi Zhu, Yibo Zhu, Xia Zhou, Ben Y. Zhao and Haitao Zheng, Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems (SenSys), Stanford, USA, November 2016.

“Differentially Private Spectrum Auction With Approximate Revenue Maximization”. Ruihao Zhu, **Zhijing Li**, Fan Wu, Kang G. Shin and Guihai Chen, Proceedings of the

15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Philadelphia, USA, August 2014.

“Adversarial WiFi Sensing”. Yanzi Zhu, Zhujun Xiao, Yuxin Chen, **Zhijing Li**, Max Liu, Ben Y. Zhao and Haitao Zheng, arXiv preprint arXiv:1810.10109.

“Understanding the Effectiveness of Ultrasonic Microphone Jammer ”. Yuxin Chen, Huiying Li, Steven Nagels, **Zhijing Li**, Pedro Lopes, Ben Y. Zhao and Haitao Zheng, arXiv preprint arXiv:1904.08490.

Abstract

Towards Practical Data-Driven Network Design

by

Zhijing Li

In today's high-bandwidth culture, network managements face a set of challenges to those of previous decades. Traditional approaches rely on the *static* and *manually configured* rules, which are proposed by the network designers through years of study. The rules fail to handle the dynamic traffic as well as the increasingly heterogeneous operating conditions that come with the modern network environment. Besides, the traditional network design paradigm is no longer effective as human understanding of the complicated network becomes much harder and costly.

Recently, *data-driven* paradigm has been proposed to fill gaps in human understanding of complicated tasks. Data-driven paradigm allows the solutions to be learnt directly from the data produced in the task. Thanks to the rapid development of network monitoring tools over the years, we are able to obtain a large amount of network measurement data that makes it feasible to apply data-driven paradigm in network management.

This dissertation aims at designing and managing the networked systems based on data-driven paradigm. However, when applying data-driven approaches into the real-world tasks, there are multiple challenges. The key contribution of this dissertation is to provide solutions to address three fundamental challenges: *First*, the measurement data may be unreliable due to factors of measurement bias, hardware, environment and human artifacts. The bad quality of the data will affect solution learning. *Second*, regardless of the data issues coming from the measurement, the real-world network data itself can be complicated (*e.g.*, bias) which requires specially designed learning algorithms to handle.

Third, we need to improve the scalability of solution (model) for better efficiency. In this dissertation, we propose practical data-driven designs for three real-world network tasks: *Radio-Frequency (RF) transmitter localization*, *network anomaly detection* and *spectrum anomaly detection*.

First, we improve the performance of RF transmitter localization by filtering out unreliable measurement data. We propose a novel application of unsupervised learning to detect hidden correlation between measurement instances, their features, and localization accuracy. We use the key features to identify the types of measurement data that correlate well with high or low prediction accuracy. By only using the measurements of high prediction accuracy, we are be able to improve the localization accuracy.

Second, we predict network issues for Network Function Virtualization (NFV). NFV is a new network architecture, and there's less domain knowledge of NFV management. We propose to learn the prediction rules from the system logs via deep learning. Since network issues are rare, we utilize Long Short-Term Memory (LSTM) to detect anomalies from the data that can potentially be used as early indicator of network issues that would typically result in trouble tickets.

Finally, we explore the design of a general, scalable system for detecting spectrum anomalies in wide-area LTE networks. We address the challenge by building context-agnostic models for spectrum usage and applying transfer learning to minimize training time and dataset constraints. The end result is a practical DNN model that can be easily deployed on both mobile and static observers, enabling timely detection of spectrum anomalies across LTE networks.

In summary, we propose a suite of algorithms and solutions driven by domain-specific insights to make data-driven designs practical and high-performing. We hope our studies could provide insights for researchers to explore new data-driven paradigms for future networking research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Globally connected networking are revolutionizing the way we live, work and play, and transforming how we communicate. Over the past two decades, we have seen tons of innovation in the devices we use to access the network, the applications and services we depend on to run our lives, as well as the computing and storage solutions that we rely on to hold data for all the services. As a result, the number of the network users steeply rises, and the Internet applications and services are becoming ever more demanding. At the end of 2017, the global internet users reached 3.4 billion [2]. According to Cisco's annual Visual Networking Index [2], the internet users will surge to 4.8 billions to 2022, with 14.6 billion connected devices, leading to 4.8 zettabytes IP traffic per year, which is over three times the 2017 rate.

To handle the exploding volumes of data traffic, complex network architecture and growing demands to improve network performance, the network operators are considering new technologies. Software-Defined Networking (SDN) [3] and Network Functions Virtualization (NFV) [4] are the two new network infrastructures to scale and support dynamic computing environments. Compared with the traditional network infrastructures which are built with custom silicon (ASICs) and purpose-built hardware, these two shift the control from hardware to software, and run on commodity, general purpose hardware. In this way, they are able to give the customers more control, allowing telcos to rapidly adjust to the dynamic markets needs in a lower-cost and scalable way.

The rapid change of the networks makes the traditional network designs less effective.

Limitations of the Traditional Network Design. Traditional network designs generally leverage the per-session[1] and manually configured rules to make decisions. For example, classic TCP congestion control relies on human-picked constants (*e.g.*, init cwnd, AIMD parameters) and is driven by feedback (*e.g.*, acks, ECN) observed within the current flow. These rules are generated through years of experience in operations. There are three main reasons of why traditional network designs become less effective to handle today's network: *First*, as the networked system is getting complicated, the heterogeneity in operating conditions increases and the decision space to optimize the service quality grows. Human understanding of the complex system becomes harder and costly. *Second*, as new technologies such as SDN and NFV coming in, we need domain knowledge of the new systems. The trial-and-error strategies used in the traditional design paradigm are fundamentally inefficient and slow in exploring the decision space. *Third*, we see growing demands to improve network performance, the conventional approaches that worked well for a "best effort" mentality are no longer good enough. Thus, we need new network designs.

Data-Driven Network Design. Data-driven paradigm has been proposed to fill gaps in human understanding of complicated tasks. Compared with the conventional paradigm, *data-driven* paradigm is directly built on the data produced in the task, instead of pure mathematical models with artificial assumptions. A key of the data-driven techniques is to use the state of art machine learning technology. Over the years, machine learning has shown advantages in dealing with complex problems such as natural

[1]A session can be a transport session (*e.g.*TCP session), or an application session (*e.g.*web session).

language processing, image recognition and bioinformatics. Given the complexity in the networked systems, it is promising to apply machine learning into the network domain for higher network performance.

Specifically, data-driven paradigm is suitable for networking due to: *First*, as the best capacities of machine learning, classification and prediction can play basic but important roles in network problems such as traffic classification [5], attack detection [6] and performance prediction [7]. *Second*, given the various network characteristics/parameters in the decision space, machine learning is able to identify the key features as the inputs, and construct a generalized model via a uniform training method for different scenarios. Recently, several studies [8, 9, 10, 11, 12] have shown the potential of applying data-driven paradigm in the network designs. For example, Remy [8] is proposed to learn to automatically generate congestion control algorithms given current traffic data. Experiments show that Remy outperforms traditional TCP in different congestion settings.

Data-driven Network Design in Real-World Network Tasks. While existing studies have shown the feasibility of applying data-driven paradigm to networked systems, it's not clear how to fully realize it in practice. This dissertation tackles the following three challenges when applying data-driven approaches to real-world network tasks:

First, the measurement data collected by network monitoring tools can be unreliable. The unreliability may come from measurement bias, hardware, environment or human artifacts. For example, the GPS signals collected from a low-end device or in an indoor environment can be extremely inaccurate. The unreliable data can play a role in tripping up data-driven conclusions. A recent study from Oxford Economics and ServiceNow [13] characterized "data gone wrong" as machine learning's biggest risk factor. In this survey, 51% of CIOs cited data quality as a substantial barrier to their company's adoption data-related services. However, most of the current data-related network services are not be able to handle the unreliable data.

Second, regardless of the data issues coming from the measurement, the real-world network data itself can be complicated. Bias is one of the most common data issues happened in the real-world networked systems. For example, in cellular networks, events of service down are relatively rare compared with normal operation events. However, these rare events are extremely critical to service providers and they want to learn the patterns of the rare events for future fast detection. Given machine learning is notorious for requiring of a large dataset for training, the small volume of the anomalous events makes it hard for machine learning to study. Thus, we need to carefully design the machine learning algorithms to deal with the rare yet critical events happened in the real-world systems.

Third, the pattern of the measured network data could vary from time to time, location to location, and device to device. Clearly, one-model-for-all cannot be applied in most cases, and we need to customize models for different context. Training models for different context requires a great amount of training data. However, data collection may take a long time, which definitely makes the data-driven approach inefficient. Data-driven network designs need to consider solution (model) scalability for practical usage.

The contribution of this dissertation is a suite of algorithms and solutions driven by domain-specific insights to make the data-driven network design practical and high-performing. Specifically, we propose practical data-driven designs for three real-world network tasks: *Radio-Frequency (RF) transmitter localization*, *network anomaly detection* and *spectrum anomaly detection.* This thesis addresses the challenge of data quality in the RF transmitter localization task, and the challenges of data bias and model scalability in the rest of the two tasks. We briefly describe the designs below.

1.1 Data-driven RF Transmitter Localization

Radio spectrum is a fixed and increasingly sought after resource. Licenses to existing spectrum bands are auctioned off by the Federal Communication Commission (FCC) for billions to cellular carriers. To open up allocated spectrum for next generation wireless devices, the FCC creates tiered spectrum access models [14, 15] where primary and secondary users can take advantage of the spectrum. Basically, primary users will have the highest priority and secondary users could use the spectrum without interfering with primary users. The new model creates a contentious environment between different types of wireless devices, and adds further urgency to the development of spectrum monitoring tools to be used for transmitter detection, location and avoidance.

Here, we focus on the transmitter localization which is a major problem in the spectrum monitoring. Typically, transmitter localization is decided by a mount of received signal measurements based on signal propagation properties. Conventional spectrum monitoring is done by the FCC by war-driving with high-end spectrum analyzers. The cost is high and the localization result is limited by the war-driving route. Our goal here is to build a better spectrum monitoring platform, which is a distributed system built on commodity smartphones and embedded low-cost spectrum sensors. In general, the network providers pay non-network users to do the measurement tasks, and the non-network users will run a crowdsourcing daemon and listen for locally broadcast measurement requests. At the same time, we require to obtain accurate localization results, even if the low-end measuring devices could have large noises.

We propose the distributed spectrum monitoring system in the first part, and show that the low-data quality of the low-end devices becomes the major blockage for accurate localization. We propose a technique to identify and overcome errors and uncertainty in the data in a statistic manner. The first part contains a relatively small-scale measurement study, in the second part, we study the quality of crowdsourced signal measurements using large-scale international public datasets. We aim at helping understand the value and limitations in crowdsourced network measurements. Our goal is to optimize the service of transmitter localization by inferring the quality of the measurement data and then doing localization with the high-quality measurement data only. What's more, we want to understand the key features of the measurements that directly affect localization results. To do that, we propose *feature clustering*, a novel application of unsupervised learning to detect hidden correlation between measurement instances, measurement characteristics, and localization accuracy. Our results identify two measurement characteristics as the key features that correlate with highly predictive measurement samples for both sparse and dense measurements respectively. We show how optimizing crowdsourcing measurements for these two features dramatically improves localization accuracy and reduces variance, and the proposed solution is able to generalize across datasets and even different networking tasks.

1.2 Data-driven Network Anomaly Detection

Recent deployments of Network Function Virtualization (NFV) [4] architectures have gained tremendous traction. Different from the traditional network architectures, NFV provides a new way to create, distribute, and operate networking services. It is the process of decoupling the network functions from proprietary hardware appliances so they can run in software on standardized hardware. NFV is able to simplify the deployment of new functionality and provide easier and more flexible management.

However, today's newly implemented virtualized network functions (VNFs) and their host commodity servers are more failure prone than dedicated hardware [4, 16, 17]. And it is challenging for NFV to do network management (*e.g.*quick troubleshooting) as the virtualization introduces more layering and reduce transparency into faults at lower layers. These downsides might negatively impact NFV deployment. For example, a critical question for NFV systems is whether they can provide availability similar to that of traditional carrier-grade systems, with up to five 9s (99.999% of uptime).

Thus, network trouble detection and prediction become extremely important for NFV. However, network troubles are relatively rare compared with the normal events happened in the network, which makes it hard to study the patterns. What's more, since the architecture is so new, people haven't got a good understanding of the system yet.

In this work, we leverage machine learning to capture the faulty patterns of the system from the system log data. We want machine learning to help us better understand the new network architecture and automatically generate rules for network failure prediction. We focus on one of the important VNF types - vPE (virtualized Provider Edge router) which is deployed on the edge of IP backbone network of a large ISP (AT&T) in the US. Given the real-world measurement data, we face three challenges for the prediction system design: *First*, since failures are relatively rare, our data is extremely imbalanced, making it very difficult to train a supervised learning model for fault ticket prediction. *Second*, since each VNF has its own specific configuration and traffic characteristics, it is likely that no single model will work well across VNFs. *Third*, periodic software updates constantly alter system functionality and traffic characteristics on the data plane. Thus we do not have the luxury of collecting a large training set to build a model for long term use. Instead, models must be built quickly using short windows of data, and deployed before they are made obsolete by the next software update or configuration change.

To address the data bias, we treat it as an unsupervised anomaly detection problem. We propose a deep neural network model utilizing Long Short-Term Memory (LSTM), to model the system log as a natural language sequence. It can automatically learn log patterns from normal execution, and detect anomalies when the log patterns deviate from the model trained from log data under normal execution. The abnormal log patterns will be treated as the trigger predictions of network faulty conditions. To address VNF diversity, we use clustering to identify VNFs with similar configuration and log behaviors, and aggregate them. To address the temporal dynamics of infrastructure changes, we use incremental training that resembles transfer learning. This helps us to quickly bootstrap a model after software updates, without incurring extended delays for collecting training data. We evaluate our methodology using the real-world measurement data. The results show that the detected anomalies can help operators to understand the faulty system, as well as to optimize trouble ticket generation and processing rules in order to enable fast, or even proactive actions against faulty conditions.

1.3 Data-driven Spectrum Anomaly Detection

Cellular providers spend billions of dollars acquiring radio spectrum for network capacity and coverage. However, spectrum management, specifically detection of faults from transmitter misconfigurations and spectrum interference, still remains a costly and ad hoc process, often involving manual diagnosis following customer complaints and operational failure logs [18]. The end result is that service outages can go for hours or even days.

What's more, these problems will grow in severity and scale in the near future. *First*, advances in both reconfigurable radio hardware [19, 20] and spectrum usage policies [15] make it easy to misuse spectrum without authorization. *Second*, cellular interfaces for IoT devices are coming. While they are optimized for the network and energy needs of IoT devices [21], adoption of these interfaces has the side effect of increasing security risks for LTE and nearby spectrum bands. A compromised device working on behalf of an attacker or botnet [22, 23] can perform jamming or denial of service attacks on cellular bands.

Clearly, cellular networks (and authorized wireless providers in general) need more robust and flexible tools to detect faults and misbehavior in radio spectrum usage, which we hereby refer to as spectrum anomalies. To do that, recent research works have proposed to integrate distributed spectrum measurement with deep learning. Basically, they demonstrated that Deep Neural Network (DNN) models are better at capturing the complex pattern of spectrum usage than other simpler models. However, their model is built for observers at static locations and only able to capture locale-specific spectrum usage pattern.

In this work, we focus on the development of practical DNN models that can be run at different observers (static or moving) to detect (and diagnose) spectrum usage anomalies from real-time spectrum measurements. *First*, we propose the spectrum usage model and anomaly detection system, using a combination of Recurrent Neural Networks (RNN) and transfer learning. The result is a set of unified regional RNN models that can be deployed at spectrum observers in each LTE cell, regardless of their mobility context (stationary, walking or driving). *Second*, we explore the feasibility of using transfer learning to minimize model training time and data requirements, so that we can quickly adopt the model to new LTE cells. *Finally*, we evaluate our RNN based spectrum anomaly detection against a range of failure modes, including failure, frequency and power misconfiguration, and intentional spectrum misuse. Our system can effectively detect but also recognize these spectrum usage anomalies. To the best of our knowledge, this is the first work to show the feasibility of building practical and general spectrum anomaly detection systems for large-scale LTE networks.

1.4 Organization

The remainder of this dissertation is organized as follows. Chapter 2 describes how we identify values for crowdsourced wireless signal measurements and use it to improve RF transmitter localization accuracy. In Chapter 3, we show how we build network trouble detector/predictor for the new network architecture - Network Function Virtualization (NFV) and help understand the system. In Chapter 4, we design practical spectrum DNN models that scale with the underlying distributed spectrum measurement system. Finally, we conclude the thesis in Chapter 5.

Chapter 2

Data-driven RF Transmitter Localization

In this chapter, we explore the data-driven designs for the networking task of RF transmitter localization. Recently, distributed/crowdsourced spectrum monitoring systems built on commodity smartphones and embedded low-cost spectrum sensors have shown to be more effective than traditional approaches. In the first part, we demonstrate the feasibility of deploying this kind of spectrum monitoring system[1] for TV band transmitter localization. We show the crowdsourced measurement can be unreliable and data quality could be a severe problem for service performance. We propose a technique to identify and overcome errors and uncertainty in the data in a statistic manner. To understand the limitation of the crowdsourcing measurement better, then in the second part, we proposed a new approach named *feature clustering*[2] to identify measurement values. Then by filtering out the bad measurements, we are be able to improve the transmitter

[1]A. Nika, Z Li, et al, Empirical Validation of Commodity Spectrum Monitoring, Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys 2016) Stanford, CA, November 2016.

[2]Z. Li, et al, Identifying Value in Crowdsourced Wireless Signal Measurements, Proceedings of the 26th World Wide Web Conference (WWW), Perth, Australia, April 2017.

localization performance. We show that this approach works well for not only cellular measurement datasets but Wi-Fi measurement dataset[3] as well.

2.1 Empirical Validation of Commodity Spectrum Monitoring

2.1.1 Introduction

Radio spectrum is a fixed and increasingly sought after resource. Licenses to existing spectrum bands are auctioned off by the FCC for billions to cellular carriers. To open up allocated spectrum for next generation wireless devices, the FCC is developing new tiered spectrum access models at multiple frequencies [14, 15]. Secondary devices can reuse "old" spectrum as long as they do not interfere with any primary (or legacy) users. Some secondary devices can become *protected entities* [15], who receive spectrum access free from interference by unlicensed secondary users.

The new model creates a contentious environment between different types of wireless devices, and adds further urgency to the development of spectrum monitoring tools to be used for transmitter detection, location and avoidance. Given the high cost in hardware and human resources for traditional spectrum monitoring, the FCC is partnering with industry to deploy online spectrum databases [24, 25] that maintain records for all primary transmitters, protected entities, and high power secondary transmitters. Secondary users can identify usable spectrum by querying the database by location and radio configuration. Spectrum databases are transparent, and easy to understand and utilize by secondary devices without paying for costly hardware to sense and detect primary

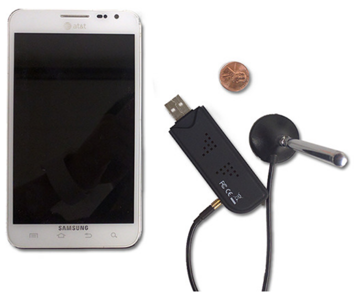
[3]Z. Li, et al, Adversarial Localization against Wireless Cameras, Proceedings of 19th International Workshop on Mobile Computing Systems and Applications (HotMobile), Tempe AZ, February 2018.

users [26].

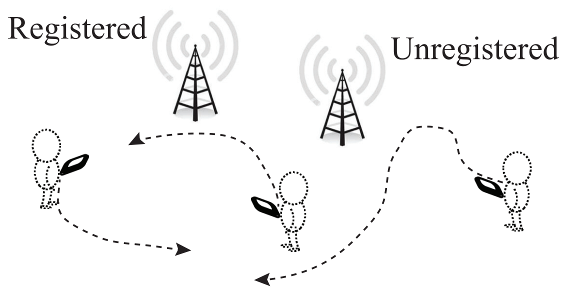
However, deploying spectrum databases does not address the difficult challenge of spectrum monitoring. Databases provide a simple way to catalog legacy wireless devices that are largely stationary, but do not simplify the task of sensing and locating new wireless devices that can be dynamic in both geographical and spectrum domains, *e.g.* portable access points for health and public safety entities, connectivity hubs for utility agencies, and interim cellular base stations to cover sudden traffic surges [14, 15]. As these devices continue to grow in number over time, the cost of spectrum databases will shift from the physical database to the cost of maintaining and updating entries to accurately reflect the frequency and physical location of active users. Exacerbating this challenge is the FCC's stringent location accuracy requirement of 50 meters [27].

A Case for Commodity Spectrum Monitoring. To date, spectrum monitoring is done by government agencies or cellular providers who perform measurements while driving around an area with specialized hardware such as spectrum analyzers. This method does not scale well for real-time, large-scale spectrum monitoring, given its costs in hardware and manpower [28]. As a result, measurement coverage is porous and sparse in many locations, making it simply impractical in less densely populated areas. Many transmitters would then evade detection, leading to large location errors in spectrum databases and undetected spectrum violations. One recent approach sought to address this problem by attaching spectrum analyzers to buses [29], but the system was severely constrained by bus routes and availability.

As flexible spectrum access policies grow in adoption around the world, it is clear that dedicated spectrum monitoring efforts will not achieve the scalability or coverage required. Instead, we believe such a system must include low-cost, commodity hardware, and leverage the growing population of active mobile devices, *e.g.* smartphones with embedded low-cost spectrum sensors. Such a distributed system, perhaps incentivized



(a) Proof-of-concept platforms



(b) Collective spectrum measurements

Figure 2.1: Commodity spectrum monitoring using collective measurements on low–cost commodity devices.

by network providers seeking to reduce spectrum monitoring costs, would have the key advantage of tying measurement density to user usage, where the system would generate the most dense measurement values and accurate sensing results in areas heavily frequented by users. Sensing results would be reported in real-time to a monitoring agency, which would process it to identify registered transmitters and locate usage anomalies.

Viability of Smartphone-based Spectrum Monitors. Deploying a highly accurate commodity spectrum monitoring system is challenging. Low-cost spectrum sensors have limited sensitivity and bandwidth compared with specialized hardware [30]. User contributed data also suffers from unpredictable mobility and human error. These sources of noise and variance can be largely addressed by taking more samples. But doing so incurs additional costs in user participation (and possible incentive costs) and energy dissipation on user devices.

The goal of our work is to validate the viability of a future measurement platform based on smartphones and commodity sensors. We believe that as demand for wireless capacity continues to grow, next generation smartphones will come embedded with flexible radios that expose more low level RF data to the OS. To validate this approach to spectrum monitoring in the absence of prebuilt devices, we are using a prototype that combines commodity smartphones and an external RTL dongle interconnected via USB, monitoring a wide spectrum range of 52-2200MHz. Our platform, while impractical for today's cellular users, provides a lower-bound for analysis on the possible efficacy of spectrum monitoring using cognitive radio embedded smartphones. We hope this study and follow-ups can provide early validation for the viability of the smartphone-based spectrum monitoring platform. Finally, we also note that US government agencies have included ultra low-cost sensors like RTL dongles as potential candidates for spectrum monitoring [31]

We implement a "proof-of-concept" sensing platform by connecting $20 USB RTL dongles to today's smartphones (Figure 2.1), and collect 17 hours of outdoor spectrum measurements on TV bands from 48 volunteers. We identify in our monitoring data multiple sources of error, including RTL hardware noise, dynamic context, user mobility bias, and RF interference. Untreated, these artifacts lead to large errors (200 meters) in transmitter location estimation. To address this, we design multiple mechanisms to effectively remove artifacts of commodity measurements, producing accurate estimates of transmitter location (<40 meters error) from a few minutes of user measurements.

Most prior work on spectrum measurements focus on spectrum occupancy and not transmitter location, and rely on specialized, high-cost hardware [32, 26, 33, 29, 34]. We show that despite a variety of error sources and hardware limitations, low-cost commodity devices can be an effective approach to baseline spectrum monitoring. We believe our findings can generalize to systems using other low-cost sensor platforms (*e.g.* [35, 36]), which might experience different levels of hardware noise, but face the same challenges from user context, mobility bias, RF interference, and energy overhead.

2.1.2 Basic Design

We seek to study the basic viability of spectrum monitoring using collective user measurements with low-cost commodity devices. In this section, we set the context for our work by describing our “proof-of-concept” measurement hardware, and our basic monitoring system.

2.1.2.1 Measurement Hardware

While today’s smartphones have multiple built-in radios, *e.g.* WiFi, Bluetooth, cellular, they only cover a very limited range of radio spectrum, *e.g.* 2.4GHz. To cover other frequencies, especially TV whitespaces (54-698MHz), our current platform leverages an external spectrum sensor.

Smartphone + $20 RTL. Our proof-of-concept platform consists of a commodity smartphone and an inexpensive Realtek dongle (RTL for brevity) [37] that connects to the smartphone via a USB cable. The RTL behaves as a spectrum sensor and collects raw spectrum usage signals; while the smartphone collects GPS data and acts as a “data processor”, translating the raw data into a data stream that is more compact and meaningful for the monitoring system. We pick RTL because of its low cost (<$20), portability (<2oz weight), wide availability, and superior frequency coverage — it operates in 52–2200MHz with a sample rate up to 2.4MHz, and transfers raw I/Q samples to the connected host on the fly.

We built an Android app to run real-time spectrum measurements, by specifying frequency range, sampling rate and time duration. During a spectrum measurement, the smartphone obtains the I/Q samples from the RTL every 1ms and calculates the corresponding RSS value. To account for the impact of channel fading, the smartphone averages over all the RSS values gathered in a measurement cycle. For our basic design, we configure the measurement cycle to 1 second, compute 1000 RSS values (one per 1ms), and record the average. Thus the app generates a (time, GPS, RSS) tuple per second, amounting to 192KB of data per hour. In this case, the RTL is always on during a spectrum measurement, mapping to a 100% duty cycle.

Later we show that our design allows the RTL on time to be significantly reduced without affecting the monitoring result. For example, we can configure each cycle to be 10 seconds but within each cycle the RTL only performs measurements for the first 1 second, mapping to a 10% duty cycle. This generates 1000 RSS values which are then averaged to produce a (time, GPS, RSS) tuple once every 10 seconds.

Measurement Precision. Prior work [30] has shown that RTLs face two key disadvantages compared with conventional spectrum analyzers. *First*, RTLs have limited sensitivity and range, failing to detect weak signals. Our experiments on TV bands show that its sensing range is roughly 150m when detecting transmitters with 20dBm EIRP (100mW power) and 1500m for those with 1W power. *Second*, RTLs have a limited sensing bandwidth of 2.4MHz compared with USRP's 20MHz. To monitor a wideband, we can segment the target band into multiple 2.4MHz sections and allow RTLs to hop across the sections. Each hop faces a small frequency switching delay (up to 50ms [30]).

2.1.2.2 Spectrum Monitoring

Leveraging user measurements, we design our spectrum monitoring system to not only record the current utilization of each spectrum band, but also identify and locate active transmitters. Transmitter localization is a basic component of spectrum monitoring, and critical to the task of interference management and dynamic spectrum allocation.

Transmitter Identification. The first step to localizing a transmitter is to identify its signal. For TV whitespaces, the FCC requires that all (high power) devices transmit identifying information conforming to a standard, allowing observers to identify the device and its location [27]. Because the identification standard is not yet defined, in this paper we consider a simple and available solution: embedding a unique transmitter identifier (defined by the FCC) inside data transmissions as *cyclostationary features* [38].

Cyclostationary features are created when signals across some sequence of radio frequency segments are repeated, generating an easy to detect energy peak in the spectral correlation function (SCF) map.

Prior work [38] developed a simple technique to achieve fine-grained control over positions of these signal peaks, encoding unique transmitter identifiers as cyclostationary features. The result is visible to any monitoring device that can sense signals on the transmitter's frequency, without decoding data.

In our system, the measurement devices can detect each feature by first capturing the RF signal on the transmitter's frequency and applying FFT to compute a normalized, discretized version of the SCF map, and then locating the feature peak using a correlation-based detection method [39]. This eliminates noise and random occurrences of cyclostationary property in the packet data itself.

Transmitter Localization Our basic design uses collective measurements from mobile users. While walking, a user uses her smartphone and RTL to collect spectrum measurements in the local area and submits the results in real-time to a monitoring agency, *e.g.* in snapshots of a few minutes each. The agency then analyzes these measurement snapshots to produce a complete view of the spectrum usage in a wide area, *e.g.*, the physical and frequency location of each detected transmitter. As users move (and start or stop their measurements), the system obtains a dynamic view of spectrum readings that scales with the number, density, and physical reach of users in the network. Our design does not require any specialized movement patterns for users.

Locating Registered Transmitters. When a registered transmitter embeds a valid ID (as cyclostationary features) in the monitored frequency, our devices can identify the feature location and extract its RSS traces. To locate a detected transmitter, we can apply a RSS-based transmitter localization algorithm on the collected data.

Locating Unregistered Transmitters. In the absence of any registration ID, the estimated transmitter location can be noisy, because the RSS can come from one or multiple transmitters. Assuming only a single transmitter is present, we can estimate its location based on RSS measurements. We leave the task of isolating and locating individual unregistered transmitters to future work.

2.1.2.3 Incentivizing User Participation

Our design assumes the agency can recruit mobile users at targeted monitoring areas. There are multiple forms of recruitment, including crowdsourcing and incentivizing in-network users [30]. Here a practical challenge is how to ensure adequate coverage. One potential solution is to leverage an ecosystem of network providers, where each provider leverages its own users (and their commodity mobile devices) to perform spectrum measurements. These service providers are active spectrum users who seek reliable spectrum usage to support/augment their services, and thus are incentivized to participate in spectrum monitoring and protect their own usage.

Spectrum measurements will come from two distinctive groups of users. First, passive measurements will be collected from each provider's own user population, by energy-efficient background app running on mobile devices. To incentivize participation, a network provider can reward participating users with small credits to network charges commensurate with actual measurements performed. Recent studies [40, 41] have shown that small monetary incentives will increase user participation in crowdsourcing tasks.

2.1.3 Quality of User-Contributed Measurements

For spectrum measurements contributed by end-users and low-cost commodity hardware, there exists obvious doubt on data quality and the impact on spectrum monitoring. In this work, we take a data-driven approach to study this concern. In the following, we first describe our efforts on collecting real world user measurements, then present our analysis on the quality of these measurements, and their impact on the accuracy of transmitter localization.

2.1.3.1 Real World Measurements

We recruited 48 volunteers via email announcements in our local area. They are between the ages of 20 and 40 and have different body shape and height. Each user was given a Galaxy SIII smartphone with our measurement app installed and an attached RTL device. In each experiment, the users walked (as they normally do) in a large neighborhood of the target transmitter, at least 200m$\times$200m in size. No further instructions were given and the users had no knowledge of the transmitter location. For our measurements, we configure the RTLs to operate in 100% duty cycle. Participants used phones provided by us and walked along areas we specified, thus no personal information was leaked.

There is no active TV whitespace transmitter in our area with ground truth location information. Thus we set up our own transmitter using a USRP N210 radio, emitting OFDM signals on a 2.4MHz band or a 6MHz band (for wideband experiments). We place the transmitter roughly 4m above the ground in each experiment. We consider two available TV whitespace bands (569MHz and 653MHz)[4] The majority of our experiments were on 569MHz. We configure our transmitter to emit at 100mW (20dBm), thus the signal detection range of a RTL is roughly 150m.

[4]We select the TV whitespace bands by querying two different whitespace databases, Spectrum Bridge [24], and Google Spectrum Database [25] to identify the available bands in our local area.

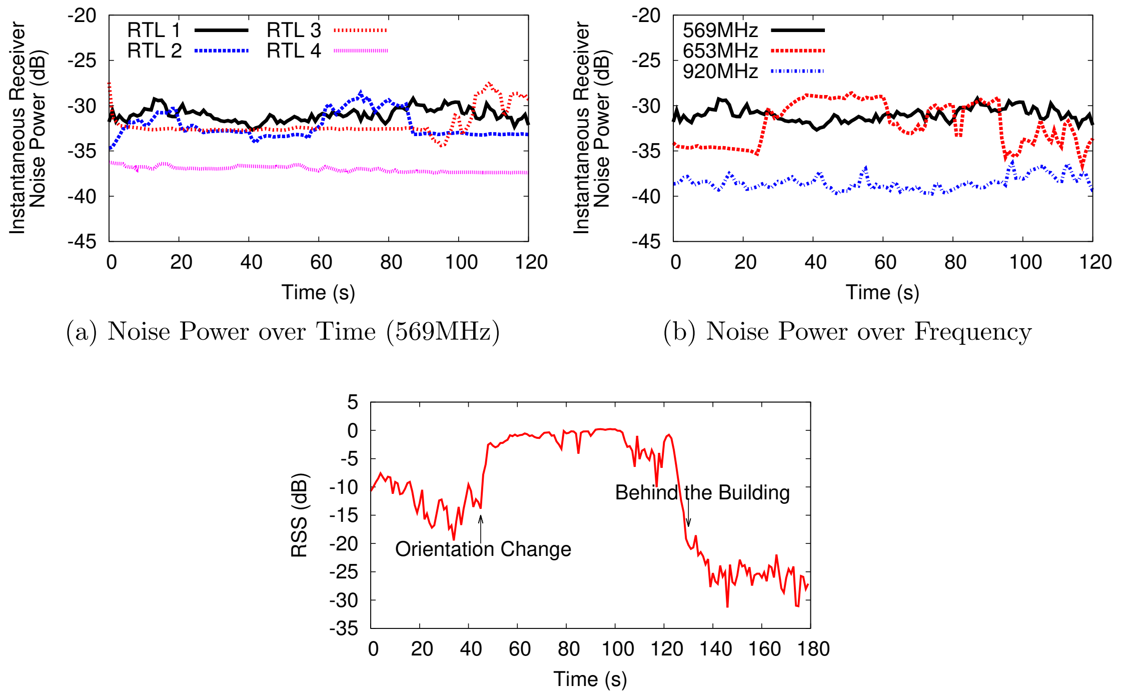
Measurement Environments. We performed extensive measurements at four outdoor environments, representing scenarios where users perform spectrum measurements during daily (walking) activities. Each experiment round involved at most 6 randomly selected volunteers.

- *Open area* – An open lawn area with minimum obstacles beside our users. Thus, RSS measurements are mostly in line of sight to the transmitter, except that the user body can block the signal.
- *Areas around buildings* – A complex area where users walk in alleys between buildings and parking lots. Both static (buildings, parked cars, trees) and mobile obstacles (pedestrians, moving cars, and bikes) were present during the measurements.
- *Downtown sidewalk* – A complex area where users walk on sidewalks between outdoor shopping stores with various obstacles (*e.g.* buildings, trees, and pedestrians).
- *Outdoor plaza* – An outdoor food court area plus a pathway to the outdoor parking lot, where users walk around during busy lunch hours.

RSS Dataset. Our measurements took place between January and March 2015 and generated a dataset of more than 17 hours of measurement data or 360,000+ (time, GPS, RSS) tuples. Among them, 1 hour of measurement data was collected by setting up two transmitters, one as a registered user who embeds its registration ID inside the transmission and another as an unregistered (and interfering) transmitter.

2.1.3.2 Data Quality Analysis

Our analysis on the RSS dataset identified a large amount of noise and inconsistency across measurements. We also identified four key sources for these artifacts. The first



(a) Noise Power over Time (569MHz) (b) Noise Power over Frequency

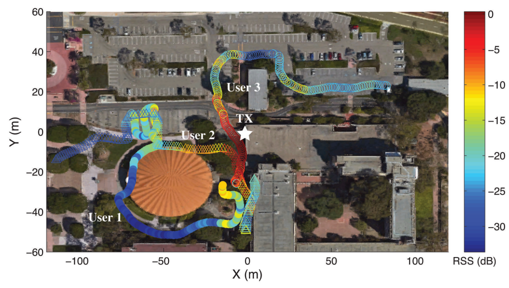
(c) Impact of Dynamic Context

Figure 2.2: Artifacts of commodity spectrum measurements. (a)-(b) RTL receiver noise varies over time and frequency and is device-dependent. (c) Dynamic user and environment context leads to sudden changes in RSS measurements.

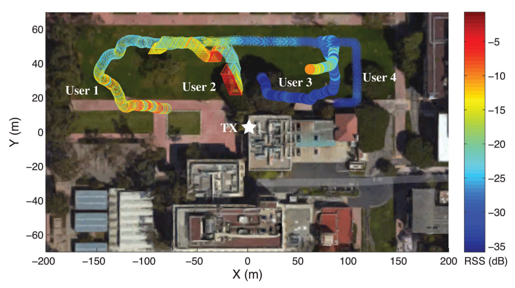
three factors are responsible for creating inconsistent and noisy RSS values, while the last factor leads to uncontrolled and biased spatial coverage.

1. RTL Receiver Noise. The low-cost RTL devices are not calibrated and have a high receiver noise figure (also reported by other studies [42]). We found that the noise level is device-dependent, and varies over time and frequency, making it very hard to model and predict. To illustrate this, Figure 2.2(a)(b) plots the measured instantaneous noise power over time for four randomly-chosen RTL devices at 569MHz, and for one RTL device at three different frequencies (569MHz, 653MHz, 920MHz).

2. Dynamic Context. Unlike war-driving with a vehicle-mounted antenna, our measurements are carried out by walking users holding smartphones and RTLs. Changes



(a) Sample Scenario A



(b) Sample Scenario B

Figure 2.3: Sample user routes and RSS measurement results. The reported noise floor is in between -30 and -35 dB.

in user movement pattern, body posture and local environment translate into random fluctuations in the data. Figures 2.2(c) shows an example where a user's reported RSS increases abruptly when her body orientation changes (from blocking the transmitter to unblocking), and later drops significantly when she walks behind a building.

3. RF Interference. When an interfering transmission is present, the measured RSS captures the aggregate power of the original signal and the interfering signal, and thus is very noisy. This is particularly harmful when unregistered transmitters "hide" behind a registered transmitter, *i.e.* using spectrum without registering as an authorized user. The corresponding RSS data will produce a wrong location of the registered transmitter.

4. Coverage Bias. Since we do not control user mobility patterns, user routes and measurement locations are uncontrolled and unpredictable. For example, Figure 2.3 plots the 3-minute routes taken by three RTLs for two scenarios, and the measured RSS values. For both routes, coverage around the transmitter is unbalanced. Such uncontrolled user routes create sampling bias, which is highly undesirable for transmitter localization.

	W.Centroid		W.Centroid P.		Gradient		Ecolocation	
	max	mean	max	mean	max	mean	max	mean
100% DC w/o int	82.6	43.1	81.7	35.4	197.1	47.8	117.3	41.2
100% DC w/ int	96.3	68.1	101.7	57.4	193.3	113.8	102.1	65.7
10% DC w/o Int	121.7	41.3	103.1	35.7	173.2	48.3	111.7	42.5
10% DC w/ int	136.2	71.2	113.6	67.5	183.4	143.8	121.9	70.1

Table 2.1: Localization error (in meters) obtained by applying conventional solutions on our RSS dataset. 100% DC references to 100% duty cycle, and w/o int refers to in absence of external interference.

2.1.3.3 Impact on Spectrum Monitoring

Together, these factors generate a considerable amount of noise and artifacts in RSS measurements. They are difficult to model and calibrate, leading to new challenges not found in conventional spectrum monitoring, *i.e.* via war-driving with high-end spectrum analyzers [33, 29]. To quantify their impact on transmitter localization, we applied five popular transmitter localization methods to our data directly. We organize the dataset into 960 snapshots of 5 minutes each, and perform localization on each instance. We pick 5 minutes because it is roughly the time a user takes to walk 300 meters, *i.e.* twice of the RTL sensing range. Thus, it represents the duration that a RTL can capture the transmitter's signal.

We consider five popular localization algorithms: centroid, weighted centroid [43], weighted centroid with Gaussian prediction [44], gradient [45], and ecolocation [46]. We also examined other well-known solutions like trilateration [47] and calibrated propagation model, and found they perform much worse. Furthermore, we study the impact of RTL duty cycle. Since our measurements were taken by RTL being always on, *i.e.* 100% duty cycle, we emulate 10% duty cycle by subsampling the dataset by a factor of 10.

Table 2.1 lists the maximum and mean localization errors under three different conditions. We omit the Centroid result since it is worse than weighted Centroid. Overall, the maximum localization error can easily reach 100–200 meters, which is too coarse for common monitoring tasks, and clearly cannot meet the FCC requirement of accuracy within 50 meters [27]. Furthermore, the accuracy varies significantly across measurement instances, again confirming the large uncertainty on the data quality. Finally, we see that the localization error degrades largely under 10% duty cycle.

2.1.4 Dealing with Noisy Data

Clearly the existing localization algorithms are unable to handle the noisy RTL measurements. To overcome this problem, we propose a robust spectrum monitoring system that combines *de-noising*, *interference removal* with *fidelity prediction*. These components allow us to remove the key noise and interference components from the RTL measurements, apply an existing transmitter localization on the cleaned data, and predict the accuracy of the localization result. As a result, our proposed solution provides three key benefits:

- **Improving localization accuracy** – By suppressing the noise and interference contribution, our solution effectively reduces the localization error.
- **Overcoming uncertainty** – By predicting the fidelity of each localization result, we enable effective decision-making in spectrum monitoring. The system can act based on fidelity levels to obtain quality results; precautionary measures may include skipping a particular measurement snapshot, or sending police devices with sophisticated hardware to do close-range verification.
- **Reducing RTL duty cycle** – By aggregating measurements across space, our solution reduces the amount of data required for accurate localization. This translates into significant reduction of RTL duty cycle, *e.g.* from 100% to 10%, with little impact on localization.

In the following, we describe the three components in detail. But to provide context, we start from briefly describing ecolocation [46], an existing transmitter localization algorithm used in our design, followed by a quick summary of our key contributions beyond ecolocation.

2.1.4.1 Background: Ecolocation

Ecolocation [46] is a widely-known algorithm for transmitter localization. The high-level idea is to capture the abstract relationship between RSS and link distance: the longer the link distance, the lower the RSS value. Unlike trilateration that represents the relationship via a propagation model, it applies a probabilistic approach to count, for each candidate transmitter location, how often the RSS-distance relationship is satisfied. The candidate location with the highest satisfaction rate is the final transmitter location.

Specifically, given a candidate transmitter location l, the algorithm calculates the distance between l and each measurement location i, referred to as $D_{l,i}$. Consider a pair of measurement locations i and j ($i \neq j$). The pair satisfies the RSS-distance relationship if one of the three conditions is met: $(RSS_i > RSS_j)$ & $(D_{l,i} < D_{l,j})$, $(RSS_i < RSS_j)$ & $(D_{l,i} > D_{l,j})$, as well as $(RSS_i = RSS_j)$ & $(D_{l,i} = D_{l,j})$. The satisfaction rate $F(l)$ of the transmitter location l is the ratio between the number of measurement location pairs that meet one of the three conditions and the total number of distinct pairs. And the final transmitter location is $TX = \operatorname{argmax}_l F(l)$.

As we will show below, our design leverages ecolocation as the underlying transmitter localization algorithm. We pick ecolocation over other candidates because it works well with small amounts of measurements and offers certain degree of robustness against noise.

2.1.4.2 Overview of Our Contributions

We make four new contributions beyond ecolocation.

- **Denoising & Localization** (§2.1.4.3) – To reduce the impact of noise, we partition the RTL data into context-based segments, apply ecolocation in each segment and aggregate the satisfaction rate across segments. Since the noise profile is much more consistent within each segment, this leads to a much more reliable estimate of the satisfaction rate, thus a more accurate localization result.

- **Predicting localization fidelity** (§2.1.4.4) – By comparing the measured satisfaction rate to the ideal value, we predict the fidelity of the localization result, and use it to aggregate localization results over time. This effectively reduces the uncertainty of the monitoring result.

- **Removing external interference** (§2.1.4.5) – By detecting and extracting the cyclostationary features of each registered transmitter, we can separate the RSS contribution of the registered transmitter and the interference, thus localizing them individually following the above two steps.

2.1.4.3 De-noising & Localization

After collecting a measurement snapshot (*i.e.*, x minutes of RSS measurements), we first partition the data into multiple segments (to isolate the noise), apply ecolocation on each segment S to obtain a per-segment satisfaction map $\{F_S(l)\}$, and then aggregate maps of multiple segments (and RTLs) into one ultimate satisfaction map $F(l) = \frac{1}{|S|}\sum_S F_S(l)$. We then determine the transmitter's location using $\{F(l)\}$.

Context based Data Segmentation. As a user walks around, the RSS value should vary smoothly over time unless the user moves behind a large obstacle or suddenly changes her body orientation (*e.g., from facing the transmitter to facing backwards and blocking the signal*). When these happen, the RSS value will experience a sudden change. With these issues in mind, we segment the data whenever two consecutive RSS observations

10% duty cycle (DC)		100% DC w/ local avg		100% DC w/o avg	
max	mean	max	mean	max	mean
52.6	18.4	44.7	17.3	71.3	21.4

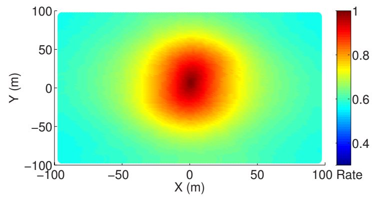
Table 2.2: Localization error (in meters) for 10% and 100% RTL duty cycle (DC).

differ by more than 8dB. We choose this threshold because our benchmark measurements show that human body blockage introduces at least 8dB loss in RSS (at least for the TV band). This value could be further optimized, which we leave to future work. Finally, we apply ecolocation on each segment S, producing a corresponding satisfaction map $F_S(l)$ per segment.

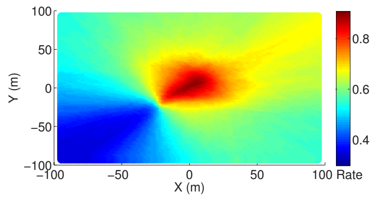
Reducing RTL duty cycle. By effectively combining data across space, we can reduce the amount of data required for accurate localization. Specifically, we find that building a good estimate of the satisfaction rate map $\{F_S(l)\}$ does not require RTL measurements at 100% duty cycle. This is because when computing the RSS and distance relationship for each location pair (i and j), i and j need to be sufficiently separated to minimize noise impact (as discussed in §2.1.4.1). Thus having fine-grained RSS data over time is only helpful when we take local average to reduce noise impact, *e.g.* averaging 10 seconds of measurements into 1 RSS value. But since each original RSS value is already an average over one second, the additional temporal average over a longer time window has limited benefits. For example, Table 2.2 lists the localization performance for 10% and 100% duty cycle values where 10% duty cycle refers to measuring every 1 second out of 10 seconds, while 100% refers to measuring in each of the 10 seconds. We see that 10% duty cycle performs similarly to 100% duty cycle with local averaging.

2.1.4.4 Predicting Localization Fidelity

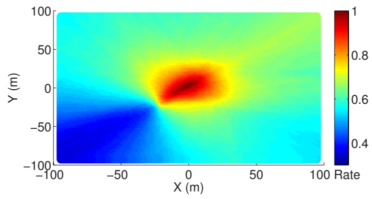
After getting a localization result (from the above step), we wish to predict the fidelity (or the level of accuracy) of the result. For this we leverage the spatial distribution of



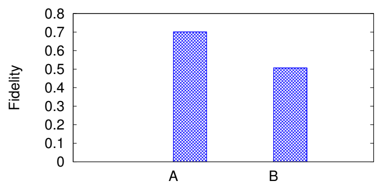
(a) Ideal Heatmap (Ideal RSS, Ideal Route)



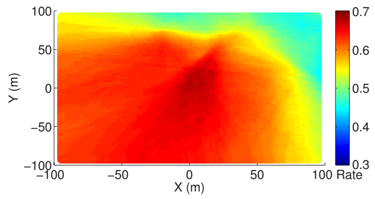
(b) Measured Heatmap (Noisy RSS, Route A)



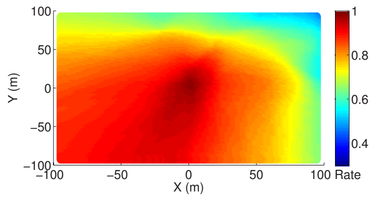
(c) Route Heatmap (Ideal RSS, Route A)



(d) Fidelity



(e) Measured Heatmap (Noisy RSS, Route B)



(f) Route Heatmap (Ideal RSS, Route B)

Figure 2.4: The satisfaction heatmaps used for transmitter localization and confidence computation.

the satisfaction rate $\{F(l)\}$.

Consider an ideal scenario where the RTL measurements are free of any noise, interference and dynamic context, *i.e.* the RSS value follows an ideal propagation model, and the measurements are evenly distributed around the target transmitter, *e.g.* a dense grid. The resulting satisfaction rate over space, hereby referred to the satisfaction heatmap, is shown in Figure 2.4(a). Here the target transmitter is located in the center (0,0), and $F(0,0) = 1$, and we assume a log-normal propagation model (for outdoor scenarios).

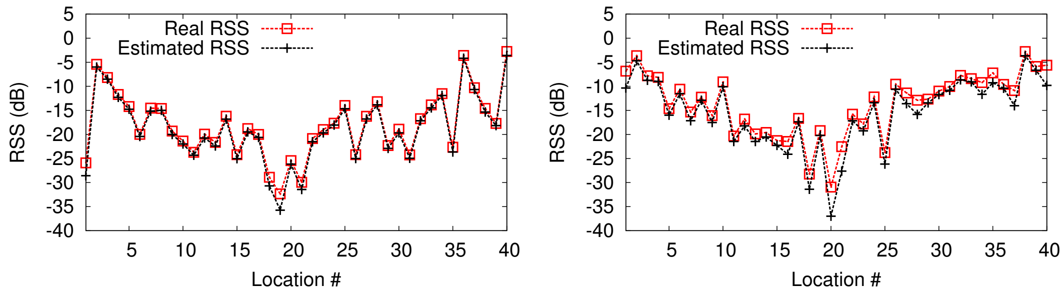
For comparison, we also plot in Figure 2.4(b) and (e) two (measured) satisfaction heatmaps built from two real RTL measurement sets on the same transmitter. For each, we center the heatmap at the estimated location. Clearly, we can observe a distinct difference between the two measured heatmaps, and their difference from the ideal heatmap (which is the same for both RTL measurements). These differences are caused by both the noise in the RTL measurements, but also the difference in user route (or coverage bias). To further illustrate these, we also plot in Figure 2.4(c) & (f) two *route* heatmaps, produced from using the actual RTL measurement locations but replacing each measured RSS with a model-generated value. By comparing these heatmaps, we can see that the difference between the ideal and route heatmaps is mostly on the heatmap structure, capturing the impact of user route (coverage bias). The difference between the route and measured heatmaps is the satisfaction value, reflecting the impact of measurement noise.

Motivated by these observations, we compute the location fidelity as the normalized cross-correlation between the ideal and measured heatmaps: $\lambda = \frac{1}{n}\Sigma_{x,y}\frac{(f(x,y)-\bar{f})(g(x,y)-\bar{g})}{\sigma_f\sigma_g}$, where $f(x,y)$ and $g(x,y)$ are the values of point (x,y) in the ideal and measured heatmaps, n is the heatmap size, $\bar{f}$ is the average of f and σ_f is standard deviation of f. Figure 2.4(d) plots λ for both snapshots, 0.7 for snapshot A (13.3m location error) and 0.52 for snapshot B (51.7m location error).

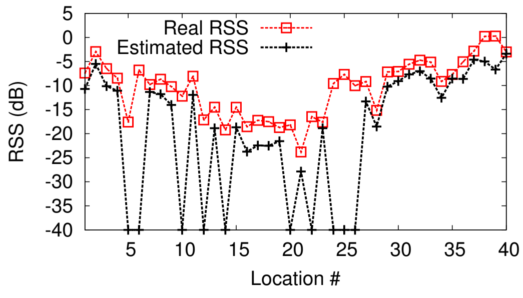
Fidelity-Guided Temporal Combining. The above discussion shows that the monitoring performance depends on the coverage of user routes which varies over time. Thus we propose a temporal combining mechanism to further improve localization. For example, we can partition a 5-minute monitoring snapshot into 3 overlapping 3-minute slots, apply the above described method to estimate the transmitter location and its fidelity value, (l_i, λ_i), $i = 1, 2, 3$. We estimate the location as $\frac{l_1\lambda_1+l_2\lambda_2+l_3\lambda_3}{\lambda_1+\lambda_2+\lambda_3}$ and the new fidelity as $\frac{\lambda_1^2+\lambda_2^2+\lambda_3^2}{\lambda_1+\lambda_2+\lambda_3}$.

2.1.4.5 Removing Interference

Under interference, our RTL devices would observe a single transmission formed by the union of the registered and unregistered transmissions. The resulting RSS captures the sum of signal and interference power. Localization based on such data is obviously unreliable.



(a) No Interference (b) Weak Interference



(c) Strong Interference

Figure 2.5: Raw received and estimated signal strength without and with interference.

Feature based Interference Isolation. At each measurement location, if the amount of interference is moderate, our RTL devices can observe a valid cyclostationary feature (related to a registration ID). From the cyclostationary feature's peak strength, we can estimate the RSS of the registered transmission that actually carries the registration ID. This allows us to separate registered transmissions from those unregistered ones, *i.e.* the interference, thus locating the registered transmitter reliably in the presence of interference.

Specifically, the strength of the cyclostationary feature that carries the valid registration ID, is proportional to $\frac{SINR}{1+SINR}$ [38]. At each measurement location, we estimate the RSS of the registered transmitter S^* from its detected feature strength s and the raw RSS S_0: $S^* = \frac{s}{\rho} \cdot S_0$, where ρ is the maximum detectable feature strength that is hardware dependent; ρ=0.99 for the RTL radios in our experiments. Our approach can detect features using very few raw I/Q samples, thus reducing RTL duty cycle has no impact here.

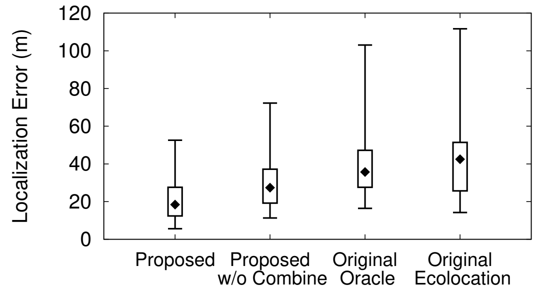
As an example, Figure 2.5 shows the measurement results at 40 locations, comparing S_0 (raw RSS) and S^* (feature estimated RSS) under three scenarios: no interference, weak interference, and strong interference. The interfering transmitter is placed 90m away from the registered transmitter, whose power level is either the same as (weak interference) or 30dB higher than the target (strong interference).

In the absence of interference, S^* is almost identical to S_0. As the interference strength elevates, S_0 grows higher than S^*, and the difference increases gracefully with the interference strength. At locations where interference is strong, we are unable to extract features and mark S^* as the noise level -40dB. Overall, as long as we can detect the feature, S^* is a reasonable replacement of S_0, the raw RSS value, which we use to locate the registered transmitter. Our results in §2.1.5 also confirm that even under strong interference, our method can still locate the registered transmitter reliably.

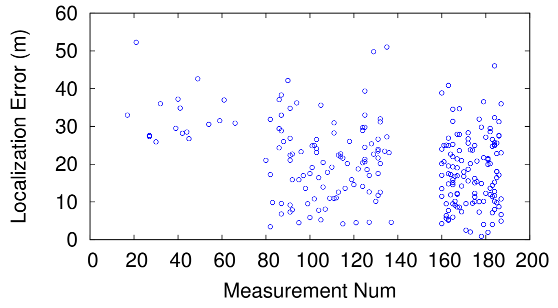
Finally, we can estimate the total interference RSS at each location as $S_0 - S^*$ and use it to approximate the interferer location assuming only one interferer is present.

2.1.5 Evaluation: Localization Accuracy

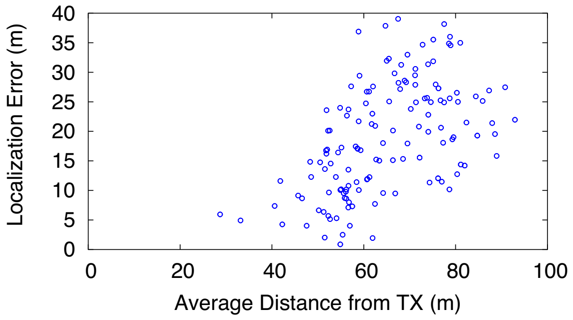
In this section, we evaluate the proposed spectrum monitoring system, focusing on the localization accuracy. We use our RTL RSS dataset described in §2.1.3, and 17 hours of measurements (16 hours without interference, and 1 hour with interference). We organize the dataset into 960 snapshots of 5 minutes each, performing localization on each snapshot. By default, we assume RTLs operate at 10% duty cycle, *i.e.* scan for 1s and then stay idle for 9s, which we emulate by subsampling our data by a factor of 10.



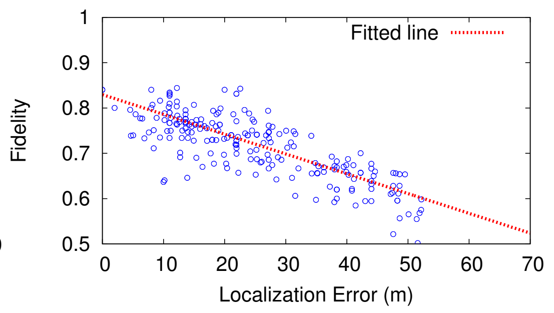
(a) Overall Accuracy



(b) Measurement Count vs. Accuracy



(c) Avg Distance from TX vs. Accuracy



(d) Fidelity Prediction

Figure 2.6: The localization accuracy of our proposed algorithm. (a) Quantiles (min, 25%, 50%, 75%, max) of the localization error across 960 snapshots. (b)-(c) Impact of measurement count and average distance from the transmitter on localization accuracy. (d) Our fidelity metric offers a good prediction of the accuracy level.

2.1.5.1 Accuracy in Absence of Interference

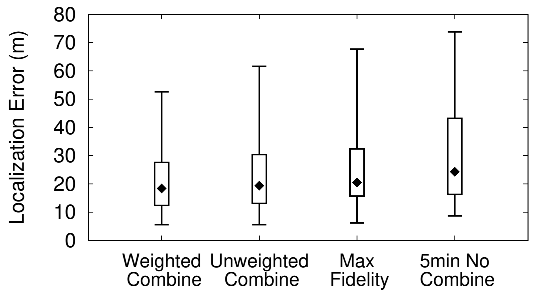
We start from the narrow band (2.4MHz) scenarios in absense of external interference. Figure 2.6(a) plots the quantile distribution of the localization error (min, 25%-tile, median, 75%-tile, max). Here we compare our proposed solution, our solution without temporal combining, the original ecolocation, and the best of the conventional localization methods, *i.e.* weighted centroid with Gaussian Prediction (as shown by Table 2.1). Compared to the two conventional localization methods, our proposed solution significantly reduces the localization error. The maximum error is bounded by 53m while the other two reach 112m and 103m (for 10% RTL duty cycle). When we increase RTL duty cycle to 100%, ours reduces to 44.8m while the best conventional method provides 82m.

Figure 2.6(a) also illustrates the breakdown of performance improvement by two components: denoising via segmentation and fidelity guided temporal combining. The difference between the original ecolocation and our proposed solution without temporal combining demonstrates the effectiveness of segmentation. The difference between our proposed solution w/ and w/o temporal combining shows the contribution of temporal combining. We can see that both components contribute to the accuracy improvement.

Performance Variance and Fidelity Prediction. We are also interested in understanding why the localization accuracy varies considerably across snapshots. For our solution, it varies between 5m to 53m, by a factor of 10. First we look at the number of measurements in the snapshot. Figure 2.6(b) shows that a snapshot with a smaller number of measurements (mostly because the number of RTLs is small) is likely to produce less accurate result, but the overall correlation is weak. A deeper analysis on the traces shows that the average distance to the transmitter is a more important factor (Figure 2.6(c)). As the user gets further from the transmitter, the impact of noise and sampling bias elevates, which degrades the localization performance.

We handle such variance by predicting the result fidelity. Figure 2.6(d) shows the predicted fidelity as a function of the localization error. We observe a good pattern between the two – higher confidence values (> 0.7) indicate more accurate localization ($< 30m$).

Effectiveness of Fidelity Guided Combining. We first consider temporal combining for narrowband monitoring. Figure 2.7 plots the quantile distribution of localization errors of the following four configurations for each 5-minute snapshot: fidelity (our proposed solution), (2) averaging the results of the 3 snapshots, (3) dividing the data into three 3-minute snapshots and selecting the localization result with the highest fidelity,



(a) Temporal Combining

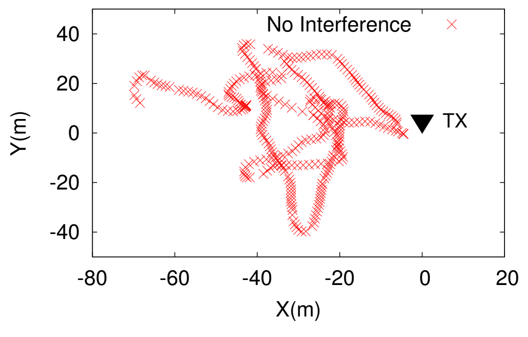
Figure 2.7: The effectiveness of our fidelity guided temporal combining.

and (4) no temporal combining. We see that weighted combining performs the best and significantly reduces the error tail. Compared with no combining, it reduces the maximum localization error from 75m to 52m. This result demonstrates the effectiveness of the fidelity guided temporal combining.

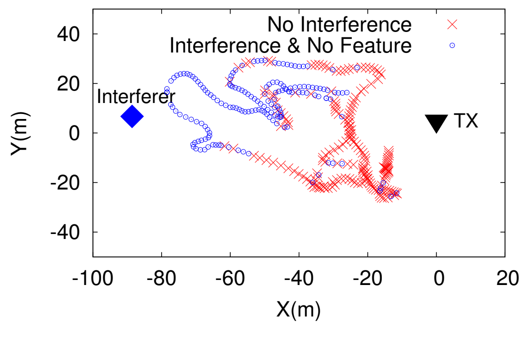
2.1.5.2 Robustness to Interference

We now consider scenarios where both unregistered and registered transmitters are present. We setup a (registered) transmitter to emit OFDM signals with embedded features. After a few minutes, we turned on an interferer that is about 90m away. Users walked by the area (without the knowledge of the two transmitters) recording raw and feature-estimated signal strength. We repeated this experiment multiple times using different power levels at the interfering transmitter.

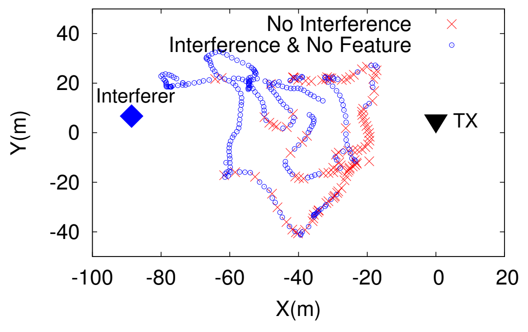
Figures 2.8 show 3-minute snapshots of two user routes in three scenarios: no interference, weak interference (the interferer has the same power level as the registered transmitter), and strong interference (the interferer's power is 30dB higher). We see that when there is no interference, the feature is always extracted (Figure 2.8(a)). Next, Figure 2.8(b) shows that under weak interference, at locations near the interferer the RTLs detect the difference between the raw RSS and the feature estimated RSS and mark



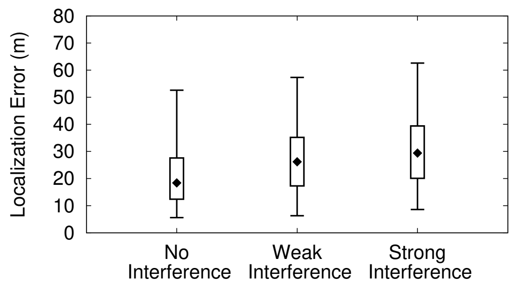
(a) No Interference



(b) Weak Interference



(c) Strong Interference



(d) Localization Result

Figure 2.8: Locating transmitters when both registered and unregistered (interfering) users are present. (a)-(c) The RTL interference measurement results under no interference, weak and strong interference. (d) The localization error when using feature-estimated RSS to locate the registered transmitter.

the locations as "interference". Finally, as the interferer becomes stronger, the number of "interference" locations increases and they are located closer to the registered user (Figure 2.8(c)).

Finally, we use the feature-estimated signal strength to locate the registered transmitter. Figures 2.8(d) shows the localization results under weak and strong interference using each 5-min snapshot. The error tail increases by 5m when the interference level increases. This is because as the interferer becomes stronger, the number of locations where a feature can be detected reduces, thus providing less input to the localization algorithm. But overall, despite strong interference, our system can locate the registered user at an accuracy similar to that of the scenario without interference.

2.1.6 Related Work

Spectrum Sensing & Measurements. Existing studies develop spectrum sensing techniques on narrowband [48, 49] and wideband signals [50, 51, 34], and improve robustness and scale using compressive sensing (*e.g.,* [52]) and collaborative sensing (*e.g.,* [32]). There are also multiple spectrum measurement platforms [32, 26, 33, 29] and some of them are used to refine TV propagation models [53, 33, 29]. Yet they all require specialized and costly spectrum analyzers (>$3500). Our work differs by using low-cost commodity radios (<$20) and collective user measurements.

Recent works implement low-cost "spectrum analyzers" using RTL and smartphone [54, 30], RTL and Raspberry Pi [35] or smartphone (WiFi) and frequency translator [36]. They also report the receiver noise caused by the low-cost radio. Another recent work [54] examines the energy consumption of RTLs when attached to smartphones. Our work differs by designing robust algorithms to deal with the noisy measurement data, sampling bias and RF interference, and by examining in detail the tradeoff between localization accuracy and energy and user cost.

Spectrum Misuse Detection. Existing studies examined spectrum misuse detection where secondary users interfere with an active primary user. They consider transmission characteristics such as the RSS distribution over space [55, 56], RSS variation [57] and physical channel features [58]. These studies either require dense sensor deployments or the availability of a sensor near each legitimate transmitter, infeasible for large-scale spectrum monitoring. These works also use data generated by propagation models. In contrast, our work uses collective measurements by low-cost RTLs to achieve real-time spectrum monitoring and transmitter location, and our evaluation is based on measurements from real-life scenarios.

Crowdsourcing Measurements. Recent efforts have leveraged crowdsourcing to collect large-scale wireless measurements, using them to characterize signal propagation and user mobility [59, 60], to understand network performance and coverage [61, 62, 63], and to improve indoor localization accuracy [64, 65]. Our work adopts a similar crowdsourcing approach but focuses on achieving real-time spectrum monitoring using low-cost commodity radios.

2.1.7 Conclusion & Future Work

In this section, we propose real-time spectrum monitoring measurements using low-cost commodity devices where measurements scale naturally with the number, density and physical reach of mobile users in the network. We use a proof-of-concept platform, *i.e.* smartphone + RTL dongle, to perform empirical validation of the platform. We show that robust data analysis can help commodity measurements overcome a variety of error sources and produce meaningful results

Moving forward, we plan to perform experiments and take a data-driven approach to multiple issues. *First*, we plan to expand our tests by locating and verifying existing TV whitespace transmitters beyond current measurements. This requires ground truth data on transmitter locations, which we hope to collect from industry partners. *Second*, we plan to expand our energy analysis using other RTL models, and optimize the measurement app to reduce energy consumption. *Third*, we will expand our work to account for false or incorrect data measurements from failures or malicious attackers, and develop mechanisms to identify and remove such anomalous reports.

2.2 Identify Values in Crowdsourced Wireless Signal Measurement

2.2.1 Introduction

As wireless networks continue to grow in size and coverage, network monitoring and management is becoming an increasingly costly and resource intensive task [66]. While it used to be a standard practice to measure wireless performance by covering an area with vehicles and specialized equipment, that is simply impractical today. Instead, companies and research firms are turning to crowdsourcing as a cheap and scalable way to perform network measurements at scale [67].

But just how reliable are these user-contributed measurements? There are obvious reasons to doubt the accuracy and the consistency of user-contributed wireless network measurements. First, unlike specialized measurement tools deployed by network providers, user-contributed measurements tend to be generated using commodity equipment with less accuracy. Second, users are often less tech-savvy, and more likely to introduce errors during operation or through user contexts (*e.g.* driving, phone in pocket). Third, crowdsourced measurements are constrained by the mobility patterns of contributing users. Therefore, measurements will follow user mobility, and are likely uneven in coverage.

With this in mind, it is critical for network providers to understand the value and limitations in crowdsourced network measurements. While crowdsourced measurements can be used for a number of management functions (*e.g.* network performance and coverage measurements [68, 69, 70], transmitter localization and radio map construction [71, 65, 72, 73], spectrum anomaly detection [74]), they are generally not amenable to quantitative analyses, because of the dearth of both measurement data and ground truth datasets.

In this work, we are taking a data-driven, quantitative approach to answering some of these questions, by focusing on the specific application of *basestation localization*. Basestation or transmitter localization is a basic operation in wireless network management, and critical to providers interested in locating misbehaving transmitters or mapping out potential holes in basestation coverage. Besides, nowadays many mobile applications rely on cell tower triangulation to determine user position [75] for lower energy consumption than GPS. However, the public sources of cell tower location are incomplete and inaccurate [72, 76]. Like other management applications, basestation localization uses received signal strength (RSS) measurements gathered by mobile devices. Unlike other applications, analyzing localization performance is tractable today, given the availability of both crowdsourced RSS datasets and ground-truth data on basestation locations.

We are interested in answering several critical questions about user-contributed signal measurements. *First*, how accurately can we locate wireless basestations using RSS measurements and known algorithms, and does accuracy correlate strongly with intuitive properties such as number or density of measurements? *Second*, can machine learning classifiers help improve location accuracy? *Third*, can we develop techniques to identify features or properties of highly accurate measurement instances, and use them to build adaptive crowdsourced measurement techniques that produce more accurate results?

Our study uses several large public datasets of crowdsourced RSS measurements gathered by user smartphone apps around the globe through the OpenCellID [72] and OpenBMap [73] projects. They are unique for two reasons: they provide raw signal measurements (compared to aggregate coverage maps), and include ground truth of real basestation locations. In total, we analyze $\sim$1M cells and 419M signal measurements. Using the ground truth data and existing localization algorithms, we first quantify the predictive quality of crowdsourced data, *i.e.* how accurately can each measurement instance predict the basestation location? We then try to identify and improve the poor localization results by applying supervised learning. Finally, we try to identify key properties of measurement instances that correlate well with localization accuracy, by taking a novel application of unsupervised learning technique we call feature clustering.

We summarize our findings as follows:

- We apply seven popular basestation localization algorithms to our ground truth datasets, and find that localization results have very high variance across a number of factors, including algorithms, datasets, and scenarios. In addition, there is a significant variance in error even across cell instances in the same dataset.

- We apply ML classifiers to improve localization accuracy. While overall accuracy is higher, error variance remains high, and our attempts to find key impactful features produce no clear results.

- We then take a novel application of unsupervised learning to identify hidden correlations in the data, which we call *feature clustering.* We define a distance metric between measurement instances based on similarity of their values in key features. Clustering the entire dataset based on pairwise distances produces key clusters that correlate features with localization accuracy of data inside them. From this, we identify RSS standard deviation and RSS-weighted dispersion mean as independent features that identify highly predictive data instances for sparse and dense measurement datasets.

- Finally, we develop an adaptive crowdsourcing technique using these two features. Applying this technique produces dramatic improvements in both increased localization accuracy and reduced variance. We also show that our results could

	# of Measurements				# of Cells			
	Germany	Poland	Russia	USA	Germany	Poland	Russia	USA
OpenCellID	390M	7.9M	4.7M	15.1M	564K	87K	157K	146K
OpenCellID-GT	13.4M	2.9M	109K	0	10.6K	21.6K	3.5K	0
OpenBMap	1.2M	58K	12.4K	317K	32.4K	1.9K	991	5.2K
OpenBMap-GT	36.6K	8.2K	1.3K	0	773	294	55	0

Table 2.3: High-level summary of OpenCellID and OpenBMap datasets. Each cell here is uniquely defined by its Cell ID.

generalize across datasets and geographic regions.

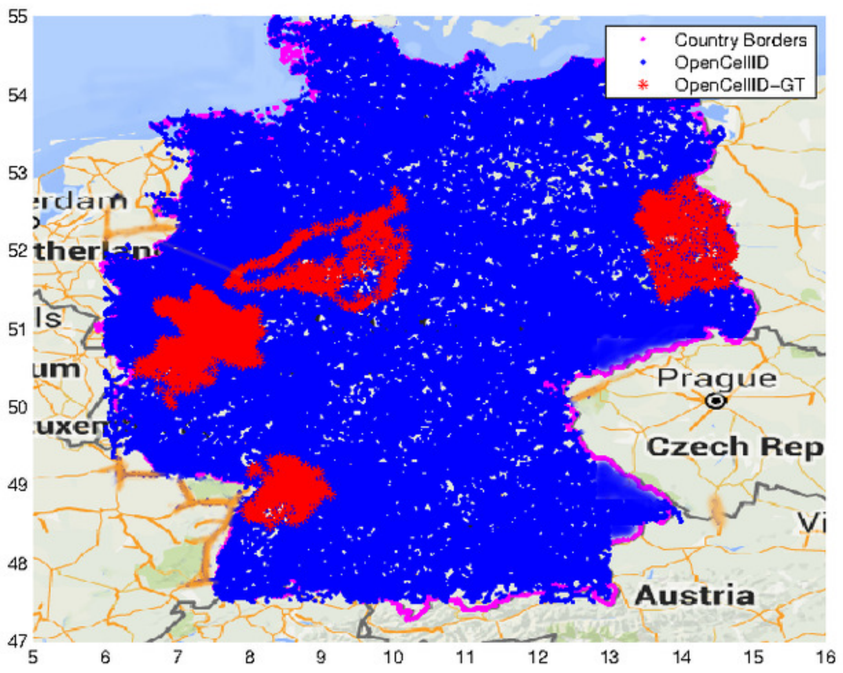
2.2.2 Datasets

Among various public datasets on crowdsourced cellular measurements [72, 73, 77, 78, 79], we use OpenCellID [72] and OpenBMap [73], for our analysis. These two datasets offer raw signal measurements, while the other ones only provide aggregated coverage maps.

OpenCellID. Created to maintain a global database of cellular basestations (identified by their Cell IDs), this dataset was collected by volunteers running a smartphone app that records information of their cellular connections. Each data entry is a single measurement at a particular time and location, containing information on the basestation (country, provider, Cell ID[5], network type) and the signal (timestamp, GPS, RSS). No user ID is included in any entry.

We group the data by country and select four countries for our analysis (Germany, Poland, Russia and USA). We pick the first three since their datasets come with the ground-truth locations of a portion of the basestations (provided by cellular service providers). We select US because it is similar to Poland in data volume. In terms of sheer volume, Germany is the largest within OpenCellID, and more than 10x larger than the #2 country; Poland and US are among top 10, and Russia ranks #20. Together, these four countries form the OpenCellID dataset, including 418M measurements

[5]Each Cell ID represents a sector of the base station, *i.e.* the antenna. A base station composed of multiple sectors will have multiple Cell IDs.



(a) Germany

Figure 2.9: The ground-truth locations of the cells covered by OpenCellID-GT, marked by red dots. They are centered around four major cities in Germany. We also include the estimated locations of the residue cells covered by OpenCellID, marked by blue dots. In the figure, the pink curve marks the country border.

and 954K cell IDs. Later in §2.2.3, we use this dataset to identify key characteristics of crowdsourced measurements.

We also create a smaller dataset OpenCellID-GT. It is a subset of OpenCellID and contains only measurements on cells with ground-truth basestation locations. The dataset includes 16.4M measurements and 35.7K cell IDs. We use it to study crowd-based basestation localization (§2.2.4, §2.2.5).

Table 2.3 summarizes the datasets in terms of the number of measurements and cells covered. We also include for each, the percentage of measurements with RSS and the percentage of cells with at least three RSS measurements per year. For a more graphical

	Weekly		**Monthly**		**Yearly**	
	GSM	**UMTS**	**GSM**	**UMTS**	**GSM**	**UMTS**
OpenCellID	16.7M	1.8M	5.4M	1.2M	725K	481K
OpenCellID-GT	883K	14.8K	292K	10.4K	44.8K	5.5K
OpenBMap	18.3K	52.9K	15.5K	43.5K	11.2K	28.3K
OpenBMap-GT	2K	201	1.6K	171	1K	109

Table 2.4: Per-cell crowdsourcing instances generated from the four signal measurement datasets in Table 1.

view, we plot in Figure 2.9 the cells covered by both OpenCellID (red & blue dots) and OpenCellID-GT (red dots). For cells without ground-truth locations, we estimate their locations as the centroid of their measurements. OpenCellID cells are spread widely in each country, except for Russia where they are concentrated in the south-west corner. In OpenCellID-GT, the cells are clustered around major cities (4 for Germany and 1 for Russia), but are widely distributed across Poland. Despite such difference, we found that the two datasets display similar structural patterns (§2.2.3).

OpenBMap. This dataset is similar to OpenCellID, but significantly smaller in size (4% of OpenCellID). Its data entry has a similar field but no ground-truth basestation locations. We will use it as a secondary dataset to verify our analysis on OpenCellID. Specifically, we consider the OpenBMap data for Germany, Poland, Russia, and USA, in 2014 and 2015. We created two datasets, OpenBMap with 1.6M measurements and 40.5K cells, and OpenBMap-GT with 46K measurements and 1.1K cell IDs. For the latter, we search for the ground-truth basestation locations from OpenCellID-GT based on their unique Cell IDs.

Per-cell Crowdsourcing Instances. From each dataset, we create *crowdsourcing instances* for each cell ID over different time windows (week, month, and year). For each window size, we partition the 2-year data into individual instances for each cell, and remove the empty instances. As a result, each cell will have multiple crowdsourcing instances for a given window size, *i.e.* up to 104 weekly instances, 24 monthly instances, and 2 yearly instances. We also group instances based on their network type (GSM, UMTS[6], LTE, CDMA etc). We find that GSM and UMTS cells dominate in both the OpenCellID (99%) and OpenBMap (95%) datasets. Table 2.4 summarizes the number of instances for these four datasets. The vast majority of basestations with ground-truth locations are GSM based, *i.e.* 88%-98.8% for OpenCellID-GT and 90% for OpenBMap-GT.

Google Basestation Location Database. We use Google's basestation location database as a reference for our localization analysis. Since 2008, Google has been collecting CellID-GPS pairs for its location-aware services [80, 81]. Also using crowdsourced measurements, they estimate each basestation (identified by the Cell ID) location as the centroid of its measurements [82]. Each estimate comes with an accuracy value ranging from 500m to 5000m, but the metric is undefined. Leveraging Google's Map Geolocation API [83], we crawled the estimated basestation locations for all the cells in OpenCellID-GT and their accuracy level.

2.2.3 Initial Analysis

We analyze our datasets to identify key properties of crowdsourced cellular measurements. We examine and compare the datasets on measurement count, spatial, and RSS statistics of per-cell measurements. We also present and contrast key results observed on OpenCellID[7] and OpenBMap, as well as consistency of results across countries and between GSM and UMTS cells.

Measurement Count. The number of measurements varies significantly across cells and across instances of each cell (between 1 and 10000), where each instance is a

[6]OpenCellID defines UMTS to include UMTS, HSPA and HSPA+.

[7]While OpenCellID-GT is a small subset of OpenCellID, our analysis shows that its structure properties are completely identical to those of OpenCellID (results omitted for brevity).

collection of measurements taken over some time window. The majority of cells have a small number of measurements – even over a year, more than 50% of cell instances have less than 20 measurements (for GSM) and 10 (for UMTS). Across countries, cells in Germany tend to have more measurements, and those in Russia have much fewer.

Spatial Distribution. To understand the spatial layout of measurements in each cell, we consider several widely used metrics [84]: average pairwise distance between measurements, diameter, dispersion (the spread of measurements around their center) standard deviation [8], and index of dispersion that quantifies the existence of clusters. For these metrics as well, we observe significant variance across cell instances. For GSM, the diameter ranges from $5mm$ (*i.e.* measurements from a single stationary user) to $68km$, while the dispersion is between $0.1km$ (measurements are near the center) and $10km$ (measurements are widely scattered and form irregular shapes). While these spatial metrics are highly correlated, they have low correlation with the measurement count $(0.02 - 0.1)$. For UMTS, while the cell size is smaller, these spatial metrics still vary widely across cell instances.

Across countries, the cells in Germany tend to have larger diameter, higher dispersion, and more clustered cells than others. For example, more than 55% of GSM cell instances have diameters larger than $4km$, which reduces to 6%, 20% and 29% for Russia, US, and Poland, respectively.

RSS. Across all measurements, RSS values were evenly distributed between (-112dBm, -51dBm[9]) for GSM and (-120dBm, -60dBm) for UMTS, and distributions were similar across four countries. Per-cell mean RSS and RSS standard deviation values both vary widely across cell instances.

Intuitively, RSS values should correlate inversely with distance to the basestation. We

[8]The standard deviation of the distance between measurement point and their centroid center, a commonly used dispersion metric.

[9]The RSS value is capped by -51dBm in all GSM measurements.

test this hypothesis using ground-truth basestation locations in OpenCellID-GT. Since basestations are generally configured with the same transmit power, we look for this relationship using the Pearson correlation coefficient γ. Ideally, γ should be close to -1. Instead, a large portion of the cells (50% for Germany, 40% for Poland and Russia) display weak correlation ($-0.5 < \gamma < 0.5$), while 10% of cells in Poland and Russia even display strong positive correlation ($\gamma > 0.5$).

This highly unpredictable relationship between RSS and distance to basestation is somewhat expected in crowdsourced measurements, since so many other factors can have strong impact on RSS values. This underscores the level of randomness present in crowdsourced measurements, and is a key reason why these datasets are less useful than controlled datasets.

User Context. 22% of OpenCellID measurements contain information on moving speed and phone direction. Our analysis on these data shows that the vast majority of measurements came from moving users. For Germany and Poland, many users were traveling at high speeds (in vehicles). The reported phone directions were uniformly distributed. Finally, 3% of measurements report estimated GPS error but the data volume is too small to offer representative results.

OpenBMap vs. OpenCellID. While much smaller in data volume, the OpenBMap datasets have similar per-cell characteristics as those of OpenCellID. The key difference is that for Germany, the measurement diameter and average pairwise distance are much smaller than those in the OpenCellID Germany datasets. As a result, the spatial properties in OpenBMaps become much more consistent across the four countries.

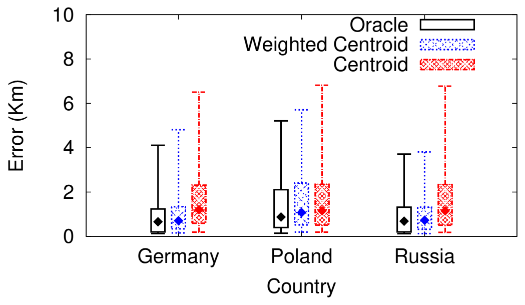
Localization Methods		**Estimated Basestation Location**
Non-RSS	Centroid (C) [43]	Geometric center of all the measurements
	Minimum Enclosing Circle (MEC) [85]	Center of the minimum enclosing circle of all the measurements
RSS-based	Weighted Centroid (WC) [43]	RSS-weighted geometric center of all the measurements
	Highest RSS [86]	Location of the measurement with the strongest RSS value
	Model-based [87, 88]	Location of the strongest RSS predicted by the calibrated propagation model
	Grid-based [89]	Center of the grid with the highest likelihood of RSS to be the strongest RSS
	Ecolocation [46]	Location with the highest value on the statistical RSS-distance relationship heatmap

Table 2.5: Summary of seven commonly-used base station localization methods.

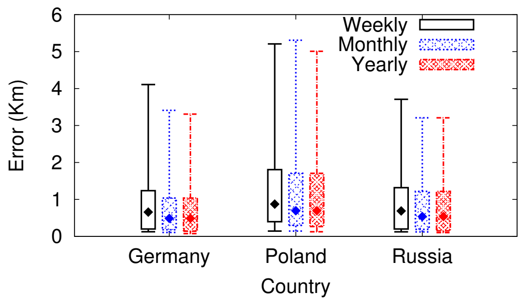
2.2.4 Localization Performance

We now examine whether crowdsourced signal measurements can be used to accurately locate basestations. Our analysis uses the OpenCellID-GT dataset, which we have shown to closely mimic OpenCellID in terms of structural characteristics. We also use the OpenBMap and Google datasets to validate our findings.

We consider seven commonly known transmitter localization algorithms, summarized in Table 2.5. Two methods (Centroid and MEC) only use spatial data, *i.e.* measurement location, and the rest five methods use both spatial and RSS data. We also consider the "Oracle" method, which, for each crowdsourcing instance, outputs the best localization result across the seven algorithms. It provides the upper bound on localization accuracy assuming one can always pick the best localization method for a given instance. We apply these algorithms on the OpenCellID-GT dataset, focusing on cell instances with at least three valid RSS measurements. We evaluate these algorithms in terms of the localization error, *i.e.* the distance between the estimated and ground-truth basestation locations. We separate our analysis for GSM and UMTS cell instances, but find that they lead to consistent conclusions. For brevity, we only show the results for GSM cells since they dominate the dataset.



(a) Weekly, Different Algorithms



(b) Oracle, Different Time Windows

Figure 2.10: (a) Comparing Centroid, Weighted Centroid and Oracle (best of 7 algorithms), in terms of quantiles (5%, 25%, 50%, 75%, 95%) of the localization error distributions. (b) The localization error of Oracle for different time window sizes.

2.2.4.1 Key Results

Across the seven algorithms, we found that no single algorithm is consistently the best, but Weighted Centroid is more likely to be the best. As an example, Figure 2.10(a) plots the distribution quantiles (5%, 25%, 50%, 75%, 95%) of the localization error based on weekly measurements, where Weighted Centroid already closely approximates "Oracle" (the best of the seven algorithms). We also observe that for 67% of the cases, RSS methods outperform non-RSS methods. As shown by Figure 2.10(a), Centroid has a much longer tail than Weighted Centroid. This means that RSS data does help localization but must be handled carefully.

Large Variance across Cell Instances. The most significant observation from our analysis is that for all the localization algorithms including "Oracle", the localization error varies significantly across crowdsourcing cell instances. For example, Figure 2.10(b) plots the quantiles of the localization error of "Oracle", for weekly, monthly, and yearly cell instances. The localization error varies significantly between $0.01km$ and $6km$. The gap between the 75- and 95-percentile values is particularly large, often more than 6x larger than the median error value. The same applies to the other seven localization

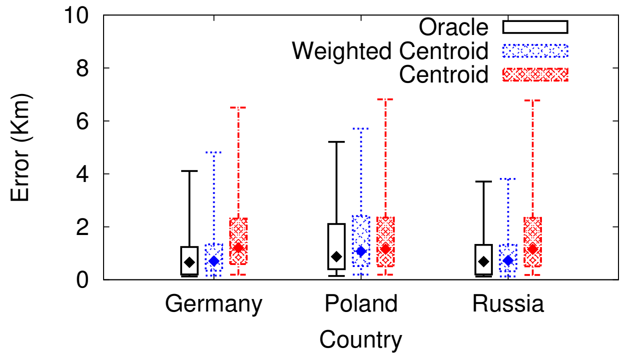


Figure 2.11: Localization error distribution across crowdsourcing instances for each individual cell.

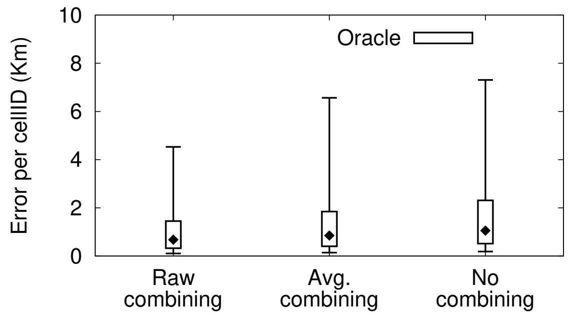


Figure 2.12: Comparing performance of sector-combining methods.

algorithms.

To understand whether the good (or poor) performance "sticks" to individual cells, we study localization error distribution across each cell's weekly crowdsourcing instances (for cells with at least 4 weekly instances). Figure 2.11 plots the quantiles of each cell's localization error, sorted by the median value. The error varies significantly across crowdsourcing instances, often by more than a factor of 10. The same applies to monthly and yearly instances (results omitted for brevity).

Together, these results show that when localizing basestations using crowdsourced signal measurements, the performance varies significantly across cells and crowdsourcing instances within each cell. Such significant variance translates into large, undesirable uncertainty in localization accuracy.

2.2.4.2 Is Sectorization Information Useful?

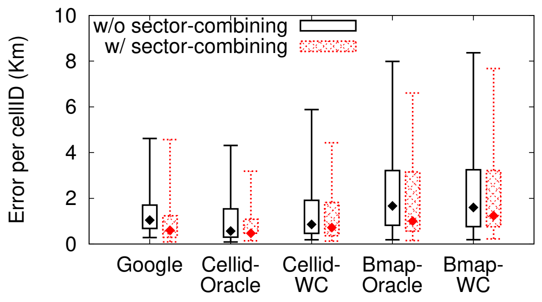
So far, we experiment on individual cells defined by their unique Cell IDs. In practice, a Cell ID often represents a single sector (or antenna) of a physical cell site, *i.e.* each physical cell can have multiple sectors. If the sectorization information, *i.e.* the set of Cell IDs associated with a single physical cell site, is available, we can perform localization by combining the measurements across their sectors. Since these sectors cover different angular regions of the cell, sector-combining can remove potential angular bias and improve localization accuracy. For our experiments, we manually[10] combine sectors by comparing the ground-truth location of each Cell ID. Together, we map 35.7k Cell IDs into 8.4k “sector-combined” cells.

We apply the same localization analysis on the combined cells. For more than 80% of cases, sector-combining improves localization accuracy by up to a factor of 10. On the other hand, sector-combining becomes harmful when noises accumulated across multiple sectors produce a strong negative impact on localization. Figure 2.12 plots the quantiles of the localization error across each Cell ID (sector) when we do not combine sectors, combine raw data across sectors, or simply average the localization result across sectors. It shows that combining raw data is more effective. But more importantly, even after combining sectors (in both ways), the localization result varies largely across cell instances.

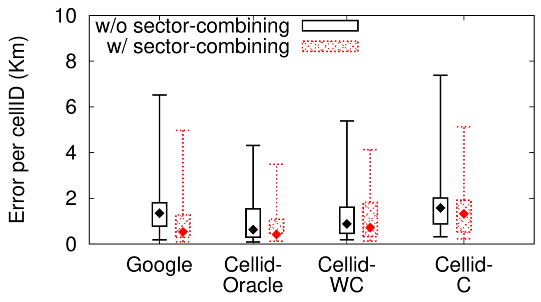
2.2.4.3 Validation via Secondary Datasets

We validate our observations using the OpenBMap and Google datasets. First, we consider the Cell IDs covered by OpenBMap-GT, and use their crowdsourced measurements

[10]Prior works [90] suggest that it is possible to discover sectorization information from Cell IDs, *e.g.* the prefix of a cell ID up to the last digit is the same for all cells belonging to the same cell tower. We do not observe such pattern in our dataset. Instead, we map two Cell IDs into one cell site if they are co-located and have the same service provider and network type.



(a) Cells in OpenBMap-GT



(b) Cells in OpenCellID-GT

Figure 2.13: Validation using the secondary datasets (OpenBMap and Google).

to perform basestation localization. Since OpenBMap has a small data volume and always offers RSS, we create cell instances using the entire 2 years of data and apply all seven algorithms. Second, for the same set of Cell IDs, we also perform localization using the 2-year data in OpenCellID-GT. We also compare their locations in the Google database to the ground-truth locations to derive the accuracy of Google's crowdsourcing approach.

Figure 2.13(a) plots the quantiles of the localization error across all the Cell IDs, using OpenCellID, OpenBMap, and Google crowdsourced measurements, with and without sector-combining. For Google, we implement the sector-combining by averaging the localization results across the sectors. While the localization accuracy varies across the datasets and algorithms, they all display significant variance across Cell IDs.

Finally, we compare Google and OpenCellID by using the Cell IDs covered by OpenCellID-GT. This is a larger dataset with 23.3K Cell IDs. Figure 2.13(b) compares the per-Cell ID localization error for Google, and OpenCellID (Oracle, Weighted Centroid, and Centroid). The Google's result is on par with OpenCellID Centroid. But overall, they again display significant variance across Cell IDs.

2.2.4.4 Summary of Observations

- **Significant performance variance across crowdsourcing instances** –This applies to multiple scenarios (w/ and w/o RSS, w/ and w/o sector-combining), localization algorithms (7 algorithms and Oracle), and datasets (OpenCellID, OpenBMap, Google).

- **Large uncertainty in localization performance** – Such significant variance translates into large uncertainty in localization performance across crowdsourced instances.

- **Information about RSS and sectorization help only if treated with care** – Extra data does not always translate into improved localization performance.

2.2.5 Predicting Localization Performance via Classification

We have observed significant variance in crowdsourced localization errors. For crowdsourced measurements to be useful, we must find techniques to distinguish "predictive" quality measurement samples from others, where "predictive" is the ability to produce localization values with low error. Our goal is to answer the question: *can we develop techniques to identify the predictive ability of crowdsourced measurement samples, and what if any "features" can help*? The most intuitive feature is the number of measurements in each cell instance, *i.e.* the more the measurements, the better the prediction. However, we confirmed across all of our datasets that measurement count is not a reliable metric, and shows no detectable relationship to localization error. The Pearson correlation coefficient between measurement count and localization accuracy lies between [-0.06, 0.02] across different scenarios and datasets.

In this section, we search for good indicators of a instance's prediction accuracy, by

Dimension	Feature Class	Details
Spatial	measurement count	
	diameter	
	clustering: index of dispersion [91]	
	clustering: nearest-neighbor index	
	minimum enclosing circle radius	
	dispersion	max, min, median, mean, StdDev, coefficient of variance
	angular coverage	
	standard deviational ellipse [92]	StdDev(major), StdDev(minor), StdDev(major)/StdDev(minor)
RSS	RSS	max, min, median, mean, StdDev, coefficient of variance
	RSS (power level, dB)	
	% of RSS (power level)> γ	γ=-55, -60, -65, -70 and -80dB
	# of RSS (power level)> γ	
Algorithm	distance between algorithm M and N's location estimates	
RSS-Spatial	RSS-weighted dispersion	max, min, median, mean, StdDev, coefficient of variance
	RSS-weighted standard deviational ellipse	StdDev(major), StdDev(minor), StdDev(major)/StdDev(minor)
	correlation between measurement distance to center and RSS	
	spatial autocorrelation [93]	
	estimated path loss exponent	

Table 2.6: Features considered in our analysis.

applying a variety of machine learning classifiers using features of the crowdsourced measurement samples to distinguish between "high quality" and "low quality" measurement instances. Not only do we seek to develop tools to identify predictive instances, but we also wish to identify key features associated with accurate measurement data samples.

2.2.5.1 Feature Selection and Training

The complex structure of crowdsourced data means that it is unlikely that the localization accuracy is controlled by a single property. Thus we consider a classifier-based approach. For a given localization accuracy requirement, *e.g.* the localization error $< x$, we seek to predict whether a crowdsourcing instance can produce localization results meeting such requirement, while identifying the key features that lead to such good (or poor)

w/o sector-combining	**w/ sector-combining**
RSS-weighted dispersion mean	RSS-weighted dispersion mean
RSS-weighted dispersion StdDev	RSS-weighted dispersion StdDev
RSS StdDev	RSS StdDev
RSS-weighted StdDev(major)/StdDev(minor) (define as *directional bias*)	Max uncovered angular
distance of localization results of Centroid and Weighted Centroid	distance gap between Centroid and Weighted Centroid (CWC)
-	RSS >-60dB

Table 2.7: Features selected by CFS.

performance.

Feature Extraction. We build four categories of features to characterize the crowd-sourcing datasets: *spatial*, *RSS*, *localization algorithm*, and *combined RSS and spatial*. The spatial features are those used by common spatial analysis [84]. The RSS features represent the statistical distribution of the RSS within each cell. The algorithm features look at the difference between results of different localization algorithms. And the combined RSS and spatial features capture the joint distribution of RSS and spatial properties, and the spatial properties of the strong measurements. Table 2.6 lists the features and the detailed descriptions are in the Appendix.

Feature Selection. Our initial feature set in Table 2.6 is large, and may contain features that are either redundant or irrelevant. To prevent overfitting, we first apply the correlation feature selection (CFS) [94] to identify a subset of relevant features for the classifier. CFS selects features independent of the classifier, and applies two criteria: the feature must be highly indicative, and must be highly uncorrelated with the features which are already selected. Table 2.7 lists the set of features selected via CFS, which are consistent across countries and time windows.

The selected feature set is dominated by RSS related features. When applying sector-combining, the feature set differs slightly. This is because combining sectors largely increases the spatial (especially angular) coverage of the measurements. As the coverage is no longer an "eclipse" away from the base station, but a complex ring structure around it, the angular coverage is more important. Finally, it is interesting to see that the distance between the localization results of Centroid and Weighted Centroid becomes a key feature. Since Centroid only focuses on spatial characteristics while Weighted Centroid utilizes both RSS and spatial properties, this feature will likely capture the complex interaction of spatial and RSS factors during localization.

Classifier Training and Testing. Using the above features, we build our classifier using multiple methods including Decision Tree, Random Forests (RF), and Support Vector Machine (SVM)[11]. For a given localization accuracy requirement, *e.g.* localization error $< x$, we prepare the training data based on the localization error obtained using a specific localization algorithm, *e.g.* Weighted Centroid. We label a cell instance whose localization error is less than x as 1 (good) and otherwise as 0 (bad). The trained classifier will output whether a testing instance is good or bad.

Following the above process, we train and test our classifier using 10-fold cross-validation[12], and report classification accuracy, precision, recall, F-measure and area under ROC curve (AUC)[13]. As expected, Random Forests produce the best classification result, since the crowdsourced data is complicated and noisy. Random Forest is better to handle noise in the data because of ensemble technique.

Table 2.8 lists the top features selected by Random Forest and their ranking in terms of the permutation importance. For comparison, we also list their ranking values computed

[11]We use the implementation of these algorthms in WEKA [95] with default parameters.

[12]The 10-fold cross-validation is a commonly used method to train classifiers and validate their prediction accuracy in practice. It is designed to limit problems like overfitting, and give an insight on how the classifier will generalize to an independent, unknown dataset. It partitions the dataset randomly into 10 equal sized (non-overlapping) subsets. Of these 10 subsets, 1 subset is used as the validation data to test the model, and the remaining 9 subsets are used to train the classifier. This process is repeated 10 times, with each of the 10 subsets used only once as the testing data.

[13]Higher AUC values indicate stronger prediction power. AUC>0.5 means the prediction is better than random guessing.

Feature	**Feature ranking methods**	
	Information Gain (CFS)	Random Forest
RSS-Weighted Dispersion Mean	0.18	0.27
RSS StdDev	0.15	0.21
RSS-Weighted Dispersion StdDev	0.14	0.18
Directional Bias	0.09	0.20
CWC Gap	0.04	0.13

Table 2.8: Feature rankings and importance.

	G-1km	G–0.5km	P–1km	P-0.5km	R–1km	R-0.5km	US-1km	US-0.5km
w/o sector-combining								
Accuracy	0.84	0.84	0.84	0.88	0.82	0.80	0.81	0.82
Precision/Recall	0.84/0.85	0.84/0.84	0.84/0.84	0.87/0.88	0.81/0.82	0.79/0.80	0.81/0.80	0.82/0.82
AUC	0.91	0.92	0.92	0.93	0.88	0.88	0.88	0.88
w/ sector-combining								
Accuracy	0.85	0.83	0.85	0.88	0.85	0.81	0.82	0.83
Precision/Recall	0.85/0.85	0.83/0.84	0.85/0.85	0.87/0.88	0.85/0.85	0.81/0.82	0.82/0.82	0.83/0.84
AUC	0.91	0.92	0.93	0.93	0.93	0.91	0.92	0.91

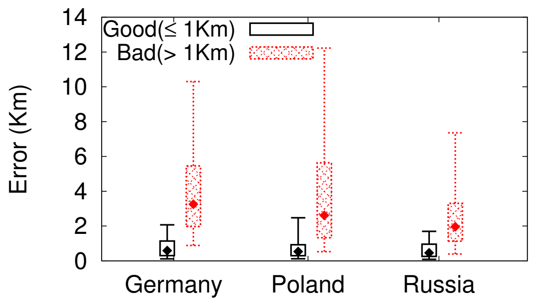
Table 2.9: Classification results for the Weighted Centroid localization algorithm for G(Germany), P(Poland), R(Russia) and US.

from CFS, *i.e.* the information gain. For both methods (CFS and Random Forest), the features have similar weights, making it hard to further locate top features among them.

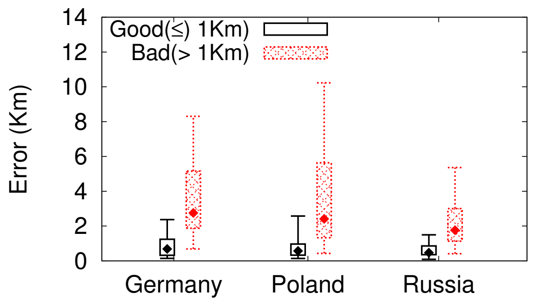
2.2.5.2 Classification Results

We now present the detailed classification results using different datasets and scenarios. We build classifiers for each country separately. Our experiment uses OpenCellID for training and testing, and OpenBMap only for testing since OpenBMap does not have sufficient data.

Table 2.9 shows the classification results when using Weighted Centroid and sector combining to perform localization, with either $x = 0.5$km or $x = 1$km as the accuracy requirement. The results are consistent across countries – the classifier has a reasonable accuracy around 85%. The performance of Russia is slightly worse, potentially due to its smaller data size (10% of the other two countries).



(a) Weighted Centroid



(b) Oracle

Figure 2.14: The distribution of the per-CellID localization error for cell instances classified as Good (<1km) and Bad (>1km).

	Poland-1km	Poland-0.5km	US-1km	US-0.5km
Accuracy	0.87	0.83	0.78	0.79
Precision/Recall/F-measure	0.86/0.87/0.87	0.83/0.84/0.83	0.81/0.76/0.78	0.79/0.78/0.79
AUC	0.93	0.92	0.88	0.88

Table 2.10: Classifier accuracy as a function of the amount of training data.

Figure 2.14(a) plots the quantile distribution of the actual localization error for the good and bad instances predicted by the classifier. Overall, we observe clear separations between the two classes. Using the classifier trained for 1km accuracy, we can identify good crowdsourcing instances that lead to no more than 3km localization error, while the majority (>75%) of these instances produce less than 1km error.

We also repeat our analysis for Oracle (the best of the 7 localization algorithms). Because the corresponding localization algorithm is unknown and will likely be much more complex, the predictability of its localization outcome reduces. As a result, the accuracy, precision, recall, f-measure reduce 2%-7% and the AUC reduces 1%-4%. Figure 2.14(b) shows the quantile distribution of the actual error for both classes. We see that the distinction between good and bad cases is more clear in 2.14(a) than 2.14(b).

Figure 2.15 shows the result of sector combining. It has slightly better performance than w/o sector combining, and the two classes can be clearly separated as well.

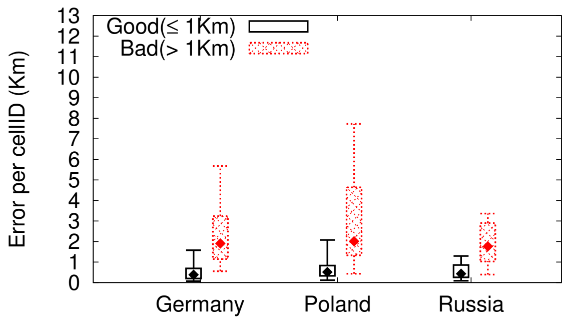


Figure 2.15: The distribution of the per-CellID localization error for cell instances classified as Good (<1km) and Bad (>1km) - Weighted Centroid, Sector-combining.

Training→Testing	Germany		Poland		Russia	
	1km	0.5km	1km	0.5km	1km	0.5km
2014 → 2015	0.83	0.80	0.84	0.86	0.80	0.77
2015 → 2014	0.84	0.82	0.84	0.85	0.81	0.79
Operator 1 → 2	0.82	0.81	0.83	0.85	0.80	0.75
Operator 2 → 1	0.81	0.80	0.82	0.83	0.81	0.79
OpenCellID → OpenBMap	0.84	0.79	0.82	0.85	0.84	0.81

Training	Germany		Poland		Russia	
Testing	Poland	Russia	Germany	Russia	Germany	Poland
1km	0.81	0.81	0.82	0.83	0.77	0.76
0.5km	0.80	0.80	0.81	0.80	0.78	0.78

Table 2.11: Cross-validation accuracy across time, cellular operators, datasets, and countries.

Impact of Training Data. We experiment with our classifier using different amount of training data. Table 2.10 lists the classification accuracy obtained using 10%, 50%, 80% and 90% of data for training. The results show that accuracy begins to reduce when there are not enough data volumes.

We also experiment with different types of training data. These include, using the data in one year to predict another year (2014→ 2015, 2015→ 2014), using the data from one cellular operator to predict another, and using the data from one country to predict another country. Table 2.11 lists the classification accuracy result. Overall, the accuracy of prediction across time and operators is on par with the original result in Table 2.9. Both Germany and Poland can well predict the other two countries, while the accuracy reduces to 0.77 when training with Russia. This is mostly because the Russia dataset is 10 times smaller than the Germany and Poland datasets.

2.2.6 Feature Clustering

Despite good classifier performance, our efforts to identify and understand key features in predicting localization accuracy using standard ML were unsuccessful. Both information gain metrics and classifiers such as Random Forests produced feature importance rankings that did not clearly distinguish between key features. While these classifiers can identify instances likely to predict location within some error, they do not shed insights on the fundamental features that are indicative of predictive measurement instances

In this section, we introduce a different approach that applies unsupervised learning to identify underlying correlations between key features and a measurement instance's predictive accuracy of Weighted Centroid. We define a distance metric that captures the similarity between key features of any two data instances. By computing the similarity metric between all pairs of instances, we can apply clustering algorithms to detect clusters of instances that capture features that tend to occur simultaneously. We call our approach *unsupervised feature clustering.*

2.2.6.1 Algorithm

Feature clustering groups data instances together based on their similarity across a small group of key features. In doing so, we are searching for possible clusters of measurement instances in the feature space, indicating a natural correlation between key features that may not be clear from other types of analysis.

By avoiding user-defined assumptions or constraints, feature clustering reveals inherent correlations between features, and allows us to identify natural combinations of features that produce highly predictive samples. Intuitively, this approach makes the assumption that a specific combination of features tends to coexist in highly predictive samples. If this assumption holds, then clusters of these features will be easily identifiable, and examining clusters will reveal key features that most strongly correlate with highly predictive measurement instances.

The process is as follows:

1. Select a small group of representative features from measurement data.
2. Define a pair-wise similarity metric between two instances based on these features.
3. Identify clusters in the measurement dataset using the similarity metric using either deterministic or heuristic clustering methods.
4. Search for correlation between identified clusters and intended outcome (in this context, prediction accuracy).
5. If strong correlation exists, use features in cluster to develop predictors for prediction accuracy.

Features and a Distance Metric. To identify a set of representative features from our measurement data, we rely on our prior results for feature selection using correlation-based feature selection (CFS) [94]. Applying CFS to a wide range of features produced a small set of features, including RSS standard deviation, RSS-weighted dispersion mean, RSS-weighted dispersion standard deviation, RSS-weighted directional bias, and CWC Gap (distance between Centroid and Weighted Centroid localization results).

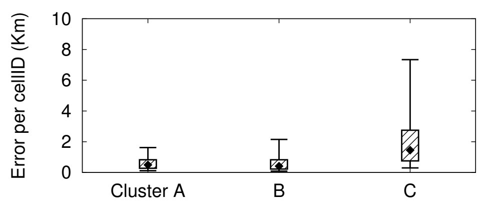
To combine the selected features into a single distance metric, we compute a five-tuple for each measurement sample (all measurement values pertaining to a single basestation). We normalize values for each feature using min-max normalization, i.e. normalized to the max value across all tuples. Finally, we generate a single distance metric by computing the (unweighted) Euclidean distance between the feature vectors of any two instances (the L2 norm of feature vectors).

Clustering. Given the distance metric, we can detect the natural clustering of measurement instances relative to our chosen features. A number of clustering algorithms are available, including hierarchical clustering [96], K-means, and METIS [97]. Since we wish to find natural correlation clusters, *i.e.* not a specific target number of clusters, we use hierarchical clustering, and optimize for modularity across all clusters. As we computed clusters for larger datasets, hierarchical clustering became a computational bottleneck. We switched to K-means for all results in this section and beyond, because it achieved nearly-identical results with an order of magnitude lower computation. We chose these clustering methods because they are commonly used and perform well in our experiments. While the choices of clustering methods could be further optimized, we leave this to future work.

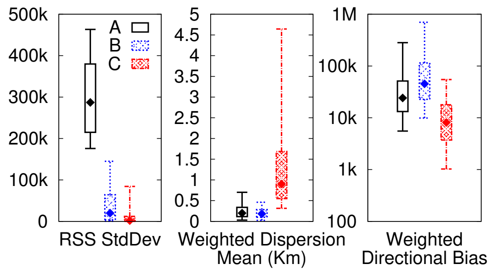
After clusters are generated, we identify the most important features by computing each feature's chi-square statistics [98] and its difference between clusters. This quantifies how different the feature's values are distributed inside and outside a cluster, and in effect captures how important each feature is to distinguishing instances in a given cluster from the rest.

2.2.6.2 Results

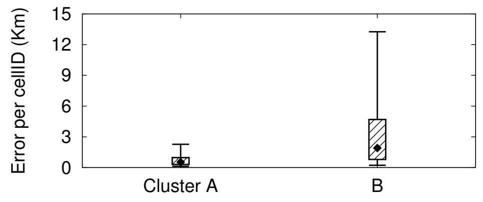
We perform clustering on measurement datasets for Germany, Poland and Russia (w/o sector combining) respectively, and plot key results in Figure 2.16. First, Figure 2.16(a)(c)(e) show that each of our three key datasets are dominated by 2 or 3 large clusters. More importantly, these clusters correlate strongly with our primary outcome, localization accuracy. In each case, one of the feature clusters identifies a group of measurement samples that produce localization results with both low error and low variance (there are two such clusters in Germany). Measurement instances in the remaining clusters produce



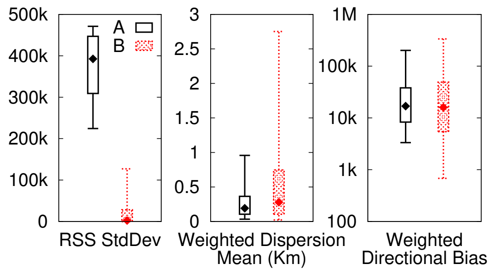
(a) Germany clusters



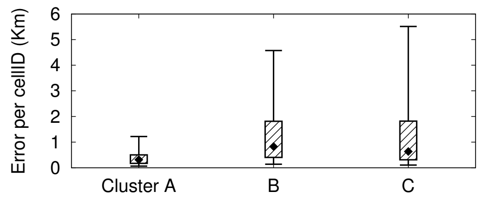
(b) Germany top featurea



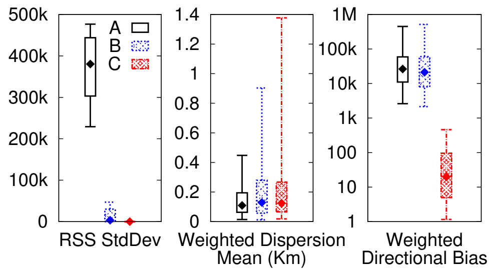
(c) Poland lusters



(d) Poland top features



(e) Russia clusters



(f) Russia top features

Figure 2.16: Clusters and top features using OpenCellID-GT (candlestick graphs).

both much higher errors and higher error variance in their localization results. These results are extremely promising, because they point to the strong correlation of these key features with localization accuracy.

A closer look at the clusters shows that the key feature distinguishing the clusters is RSS standard deviation, and RSS-weighted dispersion mean also plays a role. This is somewhat unexpected, as intuition says that signal dispersion or directional bias might be better indicators for predictive measurement instances. We plot the values for top three features: RSS standard deviation , RSS-weighted dispersion mean and RSS-weighted directional bias in Figure 2.16(b)(d)(f) (we omit the other two features because their values are similar across clusters). In all three datasets, it is very clear that the cluster with the lowest localization error and lowest error variance is defined by RSS standard deviation. In Germany, we also find a second highly predictive cluster defined by low values for RSS-weighted dispersion mean and RSS standard deviation.

Two Primary Features. These results indicate that two primary features can effectively distinguish predictive measurement instances from others. First, RSS-weighted dispersion mean is a feature that measures the mean distance from each measurement location to the estimated location of the basestation, weighted by each measurement's RSS value. So a high value is not likely generated by measurements near the basestation with high RSS values. It effectively captures the ideal scenario, where there are sufficient strong measurements close to the actual basestation. We note that this cluster only appears in our Germany dataset, which is dense, and contains a large number of measurements in urban settings. In contrast, the Poland and Russia datasets don't show this cluster in our results, because they are much sparser, and much less likely to have samples of dense measurements close to the basestation.

In the absence of well placed measurements with sufficient strong RSS values, our results show that a crowdsourced instance can produce accurate results if the measurements contain high standard deviation in RSS values. This is not an obvious result, but captures the idea that RSS measurements near the actual basestation are more diverse. The diversity comes from the signal propagation in which RSS value changes more dramatically as the receiver (smartphone) gets closer to the signal source (basestation), and user context. Regardless of whether measurements are dense (Germany) or sparse (Poland and Russia), RSS standard deviation provides a strong signal to help guide the search for predictive instances.

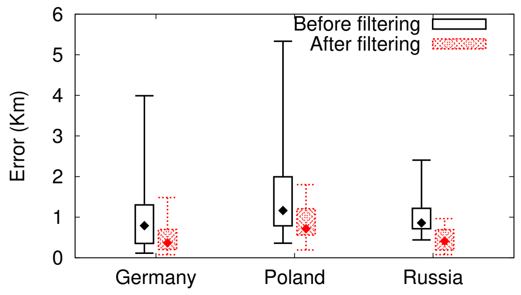


Figure 2.17: Localization results before and after filtering.

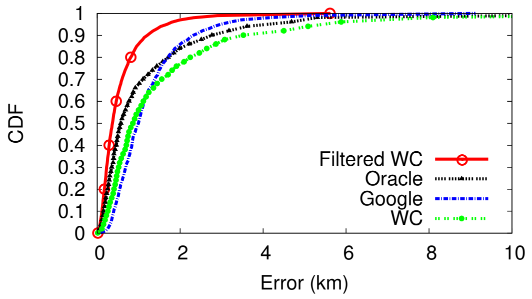


Figure 2.18: Localization results via filtering & other methods (Germany).

2.2.7 Implications & Applications

Given our insights from the previous analysis, we now consider implications on analysis of crowdsourced wireless measurements. In this section, we consider two questions. First, how can we use our insights to improve crowdsourced measurements for better accuracy? Second, we wish to test the generality of our findings by extending our approach to larger datasets in Europe and the US.

"Filtering" Crowdsourced Instances. Our key result is that RSS standard deviation and RSS-weighted dispersion mean are dominant features for detecting an instance's predictive accuracy. Generally, high RSS standard deviations and low mean RSS-weighted dispersion are both indicative of accurate localization results.

Here, we leverage these features to adaptively improve the quality of crowdsourced measurements. Our methodology is to adaptively monitor these two features as crowdsourced measurement values are gathered over time. Once a measurement instance has met either one or both of these features, we consider it sufficient. For measurement instance who has met neither target, we consider it a low-confidence instance and wait for additional measurements. We will design more complicated noise/anomaly detection in crowdsourcing measurement as a future work.

From results in Section 2.2.6, we set the bar of RSS standard deviation as 100k, and

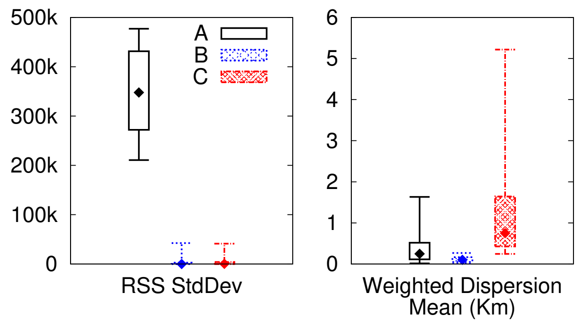
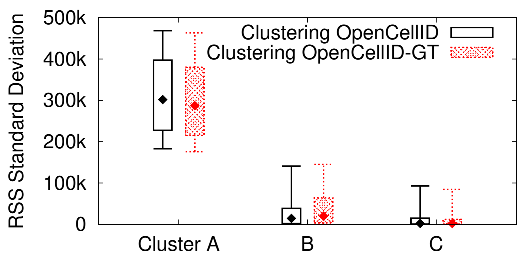


Figure 2.19: Distribution of top features for clusters in OpenCellID US.

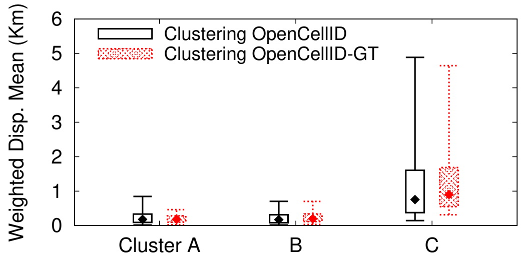
the bar of RSS-weighted dispersion mean to 0.5km. We use monthly data and look at instances that fail both targets in the three countries. We gradually add more data in the following months until the bar is reached. When getting more data, we combine the measurements of different months that maximize the RSS standard derivation and minimize the RSS-weighted dispersion mean. Figure 2.17 shows the results after this "instance filtering" process. It is clearly evident that across all countries, our filtering process dramatically lowers mean error (often by over 50%), and lowers error variance even more significantly (often by over 60%). The resulting instances are more accurate and predictable.

Next, we compare the accuracy improvements gained by instance filtering against various localization methods, for ground truth data in Germany (results for Poland and Russia are also consistent). In Figure 2.18, we use a CDF to show the accuracy improvement across all portions of the error distribution. Not only does cleaning reduce error for the bulk of all instances, but it dramatically reduces the long tail of high error instances compared to all methods.

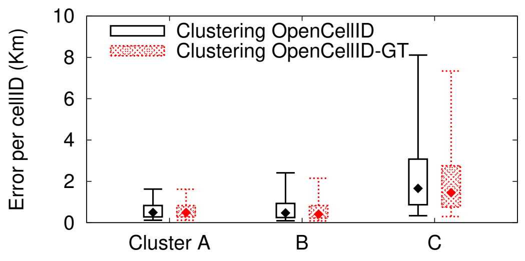
Generalizing Results. Despite these results, one might question whether our findings are a result of artifacts specific to a given dataset or location. Our hypothesis is that our conclusions are general, and high RSS standard deviation and low RSS-weighted



(a) RSS Standard Deviation



(b) RSS-weighted Dispersion Mean



(c) Localization Error

Figure 2.20: (a)(b) Feature value distributions in clusters of Germany OpenCellID and OpenCellID-GT. (c) Comparing localization error between clusters using OpenCellID-GT and OpenCellID (cells with ground truth location in each cluster are selected).

dispersion mean should be sufficient to identify highly predictive measurement instances in different settings.

To test this, we extend our methodology to different and larger datasets. In Figure 2.19, we apply our feature clustering technique to the OpenCellID US dataset. While we do not have ground truth values for this dataset, we can see that the clustered results match Germany almost perfectly (Figure 2.16(b)). There are three clusters, two clusters with low localization errors that match the high RSS standard derivation and low RSS-weighted dispersion mean features, and one cluster with high localization error.

Finally, we test our methodology on the whole Germany dataset (all measurement samples, including those without ground truth locations). In Figure 2.20, we compare the properties of the clusters produced from the whole dataset to those from the ground truth samples. The whole dataset produced the same number of clusters as the dataset of samples with ground truth, and all key properties of each cluster match the ground-truth clusters nearly perfectly.

While these results are not as strong as our earlier results because we lack ground truth, they clearly support our hypothesis that a) the clustering results and key features are not specific to the ground truth data subsets, and b) the results could generalize across datasets and geographic regions.

2.2.8 Identify Values for Adversarial Wi-Fi Localization

We further use feature clustering to identify values for Wi-Fi signal measurements. Basically, we target at improving the effectiveness of adversarial localization attacks against wireless IoT devices[14].

2.2.8.1 Adversarial Localization.

Digital homes are becoming increasingly commonplace. Some of the most popular products today are for home security. Companies like Ring and Google Nest offer cheap, high quality wireless video devices for surveillance and intruder detection. They offer users a sense of security, especially when homeowners are away from home.

As these devices gain in popularity, their widespread deployment means any security vulnerability in their design will have large-scale impact across a large user population. For wireless home cameras, one key vulnerability comes in the form of signal leakage and remote localization by external attackers[15]. From outside the home, a burglar can use wireless measurements to detect the likely location of wireless security cameras,

[14]Z. Li, et al, Adversarial Localization against Wireless Cameras, Proceedings of the 19th International Workshop on Mobile Computing Systems and Applications (HotMobile), Tempe, USA, Feb 2018.

[15]Other extensions of this attack include jamming or even modifying the camera's wireless signal. In this paper, we focus on the attack to localize the camera rather than changing its signal.

and can then plan their intrusion to avoid detection. Studies have shown that nearly 60% of home burglars will consider the presence of cameras or other video equipment when selecting targets [99]. Others show that burglars (even shoplifters) are adept at identifying and leveraging cameras' blind spots to evade detection [100, 101]. This type of attack is sometimes called *adversarial localization*, where an attacker applies localization techniques to locate third party wireless transmitters.

Adversarial Model. We consider an adversary who physically moves outside the target house/apartment/office, seeking to localize WiFi cameras behind the walls. While moving, the adversary uses a standard WiFi receiver, *e.g.* a laptop or a smartphone, to *passively* sniff transmissions from nearby WiFi devices. From the sniffed data, the adversary extracts non-payload information like MAC address, RSS and frame length. With these information, the adversary can identify WiFi cameras by directly matching their MAC addresses if she is knowledgeable enough or by analyzing traffic patterns, *e.g.* traffic volume and packet types. After recognizing each target camera, the adversary builds a signal trace per target, *i.e.* a set of tuples (time, position, RSS) along a moving trajectory, and applies TX localization to estimate the target's position.

TX Localization. We consider RSS based passive TX localization, which does not require the adversary to communicate with the victim (for stealthiness). After de-noising RSS trace using window-based averaging, we apply the log-distance path loss model to estimate the camera location. We chose this method because it is simple, widely used, and has been shown to be robust against biased spatial coverage [65]. Thus our results provide a lower-bound on the accuracy of adversarial localization using COTS WiFi sniffers.

WiFi Camera	TX Power (dBm)	Avg. Packet Size (bytes)	Packet Rate (pkt/s)
Yi Home Camera (720p, 2.4GHz)	16-18	650	60
Yi Home Camera2 (1080p, 2.4GHz)	16-18	645	75
Amcrest ProHD (1080p, 2.4GHz)	$\sim$19	1190	80
Samsung SmartCam (1080p, 5GHz)	$\sim$19	900	50

Table 2.12: WiFi cameras used in our experiments.

Environment	Adversary Behavior
Apartment	Adversary walks on the outdoor hallway.
House	Adversary walks on the lawn and sidewalk.
Office-a	Adversary walks on the indoor hallway.
Office-b	Adversary walks both inside & outside the building.

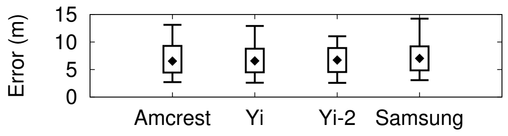
Table 2.13: Environments where we performed the attacks.

2.2.8.2 Experiment

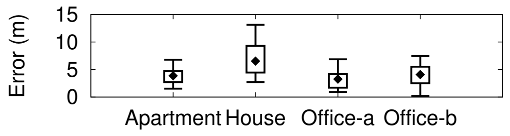
Measurements. We performed measurements on four WiFi security cameras with the highest ratings on Amazon (see Table 2.12). They use different WiFi chipsets and frequency bands (2.4GHz and 5GHz). We placed these cameras in rooms of resident houses, apartments and office buildings, and varied their locations (3 locations per room). In each experiment, the camera and the adversary were separated by walls (of different building materials). Table 2.13 summarizes the settings.

The adversary uses a laptop (Macbook Air) to sniff WiFi traffic and a smartphone (Samsung Galaxy SIII) to track moving trajectory. The adversary applies dead-reckoning on smartphone sensor data, using the accelerometer readings to track walking distance and the orientation readings to track angles. We performed experiments to confirm that the trajectory error has negligible impact on localization performance.

Our experiments were carried out by six people with different heights, weights and walking behaviors. Since they lead to consistent results, we did not differentiate them in our following discussions. Overall, our experiments produced more than 1.2k walking traces (each of 25-60 meters long), mapping to more than 2.6 million (time, position,



(a) Across camera (house environment)



(b) Across environment (mix cameras)

Figure 2.21: Localization performance in terms of quantile (5%, 25%, 50%, 75%, 95%).

RSS) tuples.

Attack Effectiveness. We quantify the accuracy of adversarial localization by *absolute localization error*, *i.e.* the distance between the estimated transmitter location and the ground truth. Figure 2.21(a) plots the quantiles (5%, 25%, 50%, 75%, 95%) of the localization error for each camera across all the walking traces. We see that the localization performance is similar across the four cameras (who use different WiFi chipsets and carrier frequencies). The median error is around 4-5 meters, which is within the room level. Yet the variance is significant, and the error can reach 12 meters. This is as expected since previous works have shown that transmitter localization is highly sensitive to measurement coverage, environmental dynamics, noises and RF interference [65].

Figure 2.21(b) then plots the localization error quantiles for different environments, mixing all the camera results. Interestingly, the accuracy varies across the environments. The error is particularly large for "house" due to more complex fading profiles and longer distance between the camera and the adversary compared to the other environments. In this case, the median error rises to 6.8 meters, while the rest three environments remain 4 meters. But a consistent trend is that the variance of localization error is large.

2.2.8.3 Advanced Attacks via Feature Clustering.

Given the large variance in localization accuracy, the basic adversarial localization faces heavy uncertainty on its effectiveness. To reduce variance, the most intuitive method is to perform multiple rounds of measurements[16] and aggregate the raw data or the localization results. We tested this approach on our dataset and found that the improvement is limited and often saturates after three rounds. Furthermore, repeatedly wandering around the target home will easily raise red flags.

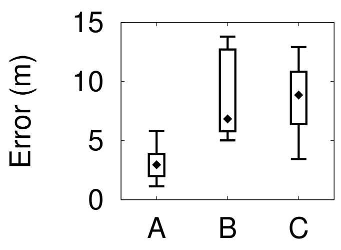
Instead, we propose to use *feature clustering* to predict the localization accuracy or *fidelity* of any measurement instance, using unsupervised learning analysis. It allows the attacker to separate high-quality measurements from noises and carry out the attack more effectively.

We start from four groups of features extracted from the raw measurement data: *packet* features on traffic statistics, *spatial* features used by common spatial analysis [84], *RSS* features on RSS statistics, and *combined* features that capture the fitting error of the localization model and the joint distribution of packet/RSS and spatial properties. We also included the environment type as a feature.

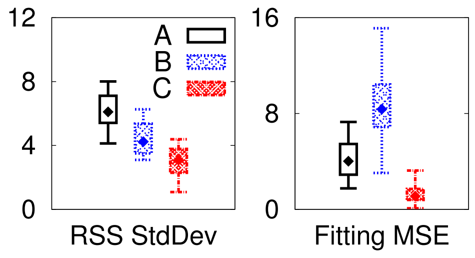
The feature clustering results are shown in Figure 2.22. We observed there are natural clusters on localization accuracy. Basically the results confirm the strong tie between feature clusters and localization accuracy. Figure 2.22(a) shows that the measurement instances in our dataset are divided into 3 clusters. Cluster A (49% of instances) produces fairly accurate localization results (2.5 meters and 5.5 meters for median and 95%-tile, respectively). Cluster B (35%) and C (16%) have relatively high localization errors (almost all $>$ 5 meters).

Figure 2.22(b) shows that two key features, *RSS standard deviation* and *fitting mean squared error (MSE)*, can be used to distinguish cluster A from the others. Measurement instances in cluster A display large RSS standard deviation but low model-fitting MSE. This is because RSS standard deviation increases as the adversary moves closer to the

[16]We assume that the attacker walks in the same area across measurement rounds, since his moving space within the camera's WiFi signal coverage is often limited, *e.g.* sidewalks, hallways etc.



(a) Clustering result



(b) Top features

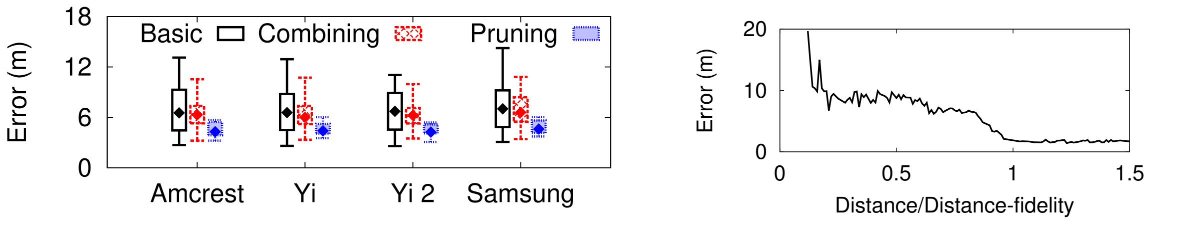
Figure 2.22: Feature clustering performance.

transmitter (camera). In this case, RSS measurements become more reliable in the presence of environmental artifacts. The next key feature is model-fitting MSE. Intuitively, this value should be small to have desirable localization accuracy. The exception is cluster C which has low values and yet bad localization results. This is because cluster C mainly consists of measurements that are far away from the transmitter and the RSS readings are *flat* across the trajectory. This type of data leads to a good model-fit, but is unsuitable for localization (since they do not capture the distance-RSS relationship).

We also observe that the clustering results and feature properties are *consistent* across all environments, cameras types and locations. Such consistency greatly facilitates the attack. An adversary can use offline, local measurements to build clusters and determine the key features used to identify high quality measurement instances. It can then derive the fidelity level instantaneously during the actual attack.

Based on this observation, we configure the feature thresholds to identify a high-fidelity measurement instance (*i.e.* those in cluster A) as RSS standard deviation above 5 and fitting MSE lower than 6. These feature thresholds do not change when we only use data from any individual environment or camera.

Attack Performance - Gain of pruning. We now show the localization performance when the adversarial uses fidelity estimation to *prune* the data, *i.e.* only using a measurement instance if its fidelity value is sufficient. As baselines, we also include results



(a) Across camera (house environment) (b) Normalized walking distance vs. error

Figure 2.23: Localization performance of advanced localization attacks.

of the basic attack and *data combining* where we aggregate three rounds of measurements since the performance saturates at this point.

We first look at the "house" environment, which has the *worst* localization performance of the four environments. Figure 2.23(a) shows the localization performance for basic attack, after combining, and after pruning. Consistently across all four cameras, pruning offers significant performance improvement, reducing the median error from more than 6.8 meters to 4.2 meters, and bounding the error by 6 meters (compared to 15 meters in the basic attack). Next, Figure 2.23(b) plots results of the four environments while mixing results of all four cameras. Again pruning leads to significant improvement in localization accuracy, reducing variance from 6.4 to 1.8. This shows that identifying useful data is much more effective than blindly adding data.

Attack Performance - Minimizing Measurement Distance. The attacker can continuously monitor the fidelity level of the current measurement. After a sufficient level of fidelity is reached, he can terminate the measurement (for stealthiness and efficiency). Figure 2.23(b) plots the localization accuracy with the moving distance. We normalize the x-axis by the distance where the required fidelity level is achieved (Distance-fidelity). We see that our fidelity estimation can accurately identify the "stopping point". Across all the high-fidelity instances (766), the min, median, max of Distance-fidelity are 6.1, 23.9, and 53.2 meters, respectively. Similarly, the attacker can use fidelity to terminate "unsuccessful" measurements.

Together, our experiments on the advanced attacks show that adversarial localization is highly effective against WiFi cameras (behind the walls). By walking around the building/house briefly, the attacker can estimate the locations of indoor WiFi cameras with room-level accuracy (median of 2.7 meters), and identify unreliable measurements.

2.2.9 Related Work

Localization using Wardriving. Initial studies explored wardriving as an approach to collect RSS measurements for localizing WiFi Access Points (AP) [43, 45, 102, 103] or cellular towers [104, 86, 90]. Kim et al. [102] concluded that state-of-art localization algorithms can produce erroneous results and this will cause inaccurate estimates of WiFi coverage. Yang et al. [90] studied the accuracy of cell tower localization using wardriving data and showed that frequency, antenna height, and propagation environment make cell tower localization different from WiFi AP localization.

Localization using Crowdsourcing. Since wardriving is cumbersome and does not provide large-scale coverage, recent studies leverage crowdsourcing for indoor localization of WiFi APs [71, 105, 106] or outdoor cell tower localization [107, 108, 109]. [107] examined several localization algorithms using only 950 measurements and showed that the grid-based approach is the best. [108] studied cell tower localization using the OpenCellID dataset and validated the results with data from only 250 users. [109] applied different localization algorithms on a small portion of OpenCellID dataset. Unlike prior studies, we examine and compare seven popular localization algorithms on two *large-scale* datasets and show that there is no algorithm that performs consistently the best. We also examine the key factors that lead to such performance variance.

Quality of Measurements. A few works addressed issues and challenges in crowdsourcing measurements. [69, 110] studied the impact of user context in crowdsourcing based cellular network measurement systems. [111] considered the problem of crowdsourced measurement distribution and data density in network coverage prediction. Li et al [65] identified that data density and environment diversity have major impact on indoor WiFi localization. In contrast, our work examines the quality issues of outdoor cellular crowdsourced measurements using large-scale datasets, focusing on basestation localization. We found that data density does not matter much to localization results. [112] investigated ways to identify true information and reliable users in real-world crowd sensing applications like air quality sensing. They require users to take measurements at the same locations which is not practical in our scenario.

For applications like web page mining, existing works (*e.g.* [113, 114, 115]) tried to remove noise and anomaly in data. Our work differs by providing a systematic framework to examine the key characteristics of crowdsourced cellular measurements and to quantify the usability of this data for basestation localization. *To the best of our knowledge, we are the first to provide a comprehensive study on the usefulness of crowdsourced wireless measurements.*

2.2.10 Conclusion

Our work analyzes the value of large user-contributed signal measurements in the context of basestation localization, using large-scale RSS datasets from OpenCellID and OpenBMap. We find that even machine learning techniques cannot reduce the variance in localization results, nor can they identify key features (RSS StdDev, RSS-weighted dispersion mean) that correlate strongly with highly predictive data instances. Instead, we apply a *feature clustering* technique to detect natural correlation patterns between measurement features, and use them to identify types of measurement data that correlate well with high or low prediction accuracy. We show that these clustering results are general across datasets, and that we can dramatically improve localization results using our identified features. We hope these results shed light on other types of crowdsourced measurements, and will test the applicability of this approach to other applications in ongoing work.

Chapter 3

Predictive Analysis in Network Function Virtualization

Recent deployments of Network Function Virtualization (NFV) architectures have gained tremendous traction. While virtualization introduces benefits such as lower costs and easier deployment of network functions, it adds additional layers that reduce transparency into faults at lower layers. To improve fault analysis and prediction for virtualized network functions (VNF), we envision a runtime predictive analysis system that runs in parallel with existing reactive monitoring systems to provide network operators timely warnings against faulty conditions. In this chapter, we propose a deep learning based approach[1] to reliably identify anomaly events from NFV system logs, and perform an empirical study using 18 consecutive months in 2016-2018 of real-world deployment data on virtualized provider edge routers. Our deep learning models, combined with customization and adaptation mechanisms, can successfully identify anomalous conditions that correlate with network trouble tickets. Analyzing these anomalies can help opera-

[1]Z. Li, et al, Predictive Analysis in Network Function Virtualization, Proceedings of 18th ACM SIGCOMM Internet Measurement Conference (IMC), Boston, USA, Nov 2018. This is a collaboration work with AT&T Labs Research.

tors to optimize trouble ticket generation and processing rules in order to enable fast, or even proactive actions against faulty conditions.

3.1 Introduction

Recent deployments of Network Function Virtualization (NFV) architectures [116] have gained tremendous traction. NFV allows network functions previously handled by hardware to be implemented as software running on commodity servers. Its advantages include simplifying deployment of new functionality, easier management through hosted VMs, and lower costs from using commodity hardware. The downsides are that 1) today's newly implemented virtualized network functions (VNFs) and their host commodity servers are more failure prone than dedicated hardware [4, 16, 17], and 2) virtualization introduces more layering and less visibility into lower layer events, *e.g.* faults. These downsides might negatively impact NFV deployment. For example, a critical question for NFV systems is whether they can provide availability similar to that of traditional carrier-grade systems, with up to five 9s (99.999% of uptime) [117].

In this paper, we describe our efforts to predict network failures and reduce downtime on one of the largest known NFV deployments to date, deployed on the edge of IP backbone network of a large ISP in the US. We focus on one of the important VNF types - vPE (virtualized Provider Edge router). We explore the design and performance of a system that would allow us to identify potential signatures for predicting trouble tickets in near-real-time, by applying a combination of deep learning models (LSTMs), model customization and sharing via transfer learning to syslogs.

While applying machine learning (including deep learning models) to failure prediction itself is not new [118, 119, 120], our work faces a unique combination of three challenges. First, because failures are relatively rare, our data is extremely imbalanced, making it very difficult to train a supervised learning model for fault ticket prediction. Second, since each VNF has its own specific configuration and traffic characteristics, it is likely that no single model will work well across VNFs. Third, periodic software updates constantly alter system functionality and traffic characteristics on the data plane. Thus we do not have the luxury of collecting a large training set to build a model for long term use. Instead, models must be built quickly using short windows of data, and deployed before they are made obsolete by the next software update or configuration change.

Our solution includes several techniques as follows:

- To address the data imbalance, we use an unsupervised anomaly detection approach to train a Long Short-Term Memory (LSTM) network [121] model with "normal" logs. Abnormal log patterns trigger predictions of network faulty conditions.
- To address VNF diversity, we use clustering to identify VNFs with similar configuration and log behaviors, and aggregate them (treat them as a single unit with the combined syslogs).
- To address the temporal dynamics of infrastructure changes, we use incremental training that resembles transfer learning. This helps us to quickly bootstrap a model after software updates, without incurring extended delays for collecting training data.

We evaluate our methodology using network trouble tickets collected over a 18-month period on vPE routers deployed in production environments. Our evaluation results demonstrate that syslog anomalies often occur before network trouble tickets are generated. We can filter through these anomalies to identify any potential early warning signals or predictive signatures.

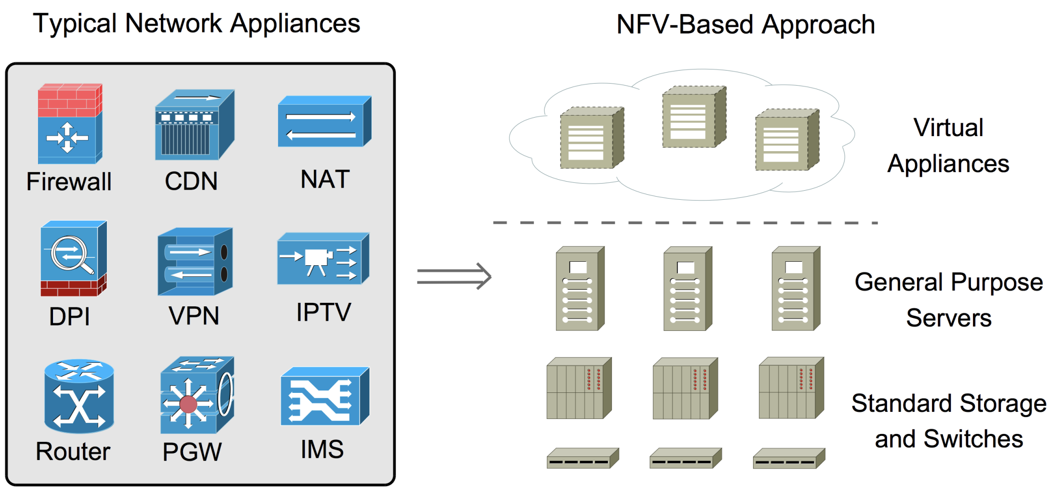


Figure 3.1: From dedicated hardware-based appliances for network services, such as firewall, Content Delivery Network (CDN), Network Address Translation (NAT), Deep Packet Inspection (DPI), Virtual Private Network (VPN), IPTV, router, Packet Data Network Gateway (PDN-GW or PGW) and IP Multimedia Subsystem (IMS), to software-based NFV solutions.

3.2 Network Function Virtualization

NFV transforms how network operators architect their infrastructure by leveraging the full-blown virtualization technology to separate software instance from hardware platform and by decoupling functionality from location for faster networking service provisioning [122]. Figure 3.1 shows NFV implements network functions through software virtualization techniques and runs them on commodity hardware (*i.e.*, industry standard servers, storage and switches).

3.2.1 NFV Architectural Framework

The NFV framework consists of three main components. We illustrate the high-level architectural framework of NFV in Figure 3.2.

Virtualized network functions (VNFs) are software implementations of network func-

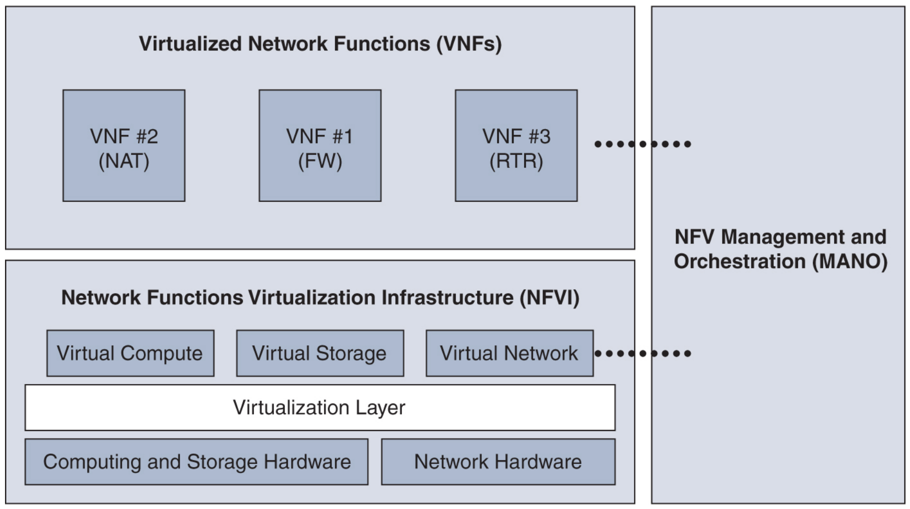

Virtualized Network Functions (VNFs)
VNF #2 (NAT)
VNF #1 (FW)
VNF #3 (RTR)
NFV Management and Orchestration (MANO)
Network Functions Virtualization Infrastructure (NFVI)
Virtual Compute
Virtual Storage
Virtual Network
Virtualization Layer
Computing and Storage Hardware
Network Hardware

Figure 3.2: High-Level ETSI NFV Framework [1].

tions that can be deployed on a network functions virtualization infrastructure (NFVI). The network service being virtualized may be implemented through a single VNF, or it may require multiple VNFs. When multiple VNFs are implementing the network service, it is possible that some of the functions have dependencies on others, in which case the VNF needs to process the data in a specific sequence (referred as service chaining).

Network functions virtualization infrastructure (NFVI) is the totality of hardware and software components that build the environment where VNFs are deployed. NFVI consists of three distinct layers: physical infrastructure, virtualization layer and the virtualized infrastructure. The virtualization layer abstracts the physical resources and anchors the VNFs to the virtualized infrastructure. It ensures that the VNF lifecycle is independent of the underlying hardware platforms. This type of functionality is typically provided in the forms of virtual machines and their hypervisors. In an NFV context, it is the responsibility of the hypervisor (or the underlying host together with its hypervisor) to perform the hardware failure detection.

Network functions virtualization management and orchestration architectural framework (NFV-MANO) is defined as a separate block in the architecture, which interacts with both the NFVI and VNF for management.

Though the NFV framework is well established, the standardization of NFV building blocks is an ongoing effort. In this paper, we focus on the failure management, which should be one of the critical functions in NFV-MANO.

Current NFV Implementation: Instead of deploying network functions spanning across physical and virtual host boundaries, the current deployment uses the simplest approach by installing it into a virtual machine (VM) image and executing it on virtual resources provided by the hypervisor.

3.3 Initial Analysis

Using data from a real-world NFV deployment, we study different types of network failures and their spatial and temporal patterns. We also examine patterns of the syslogs at the VNF layer, which we will use to predict network failures.

3.3.1 Datasets

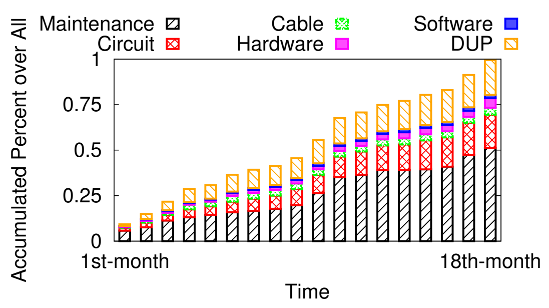
Our dataset includes both network trouble tickets and VNF syslogs collected from 38 vPEs (virtualized Provider Edge routers), deployed by a tier-1 ISP's backbone network, over a time period of 18 months. vPE degradation can cause service impairments on customer networks. Predicting these trouble events allows operators or closed-loop automations to trigger mitigation actions prior to each event and help minimize its impact.

Network Trouble Tickets. Trouble tickets capture actionable network events. Each ticket includes the time of occurrence, the root cause, and the ticket duration. Basically when a service anomaly event is detected either by various monitoring tools or customer complaints, a trouble ticket is generated and sent to the network operators. The network operators will then look into this ticket and determine whether the service deterioration was indeed caused by the suspected root cause, and record this event in a failure ticket. Thus the tickets can be used the ground-truth network failures in our case. Our dataset includes the entire set of trouble tickets at these 38 vPEs, a total of a few hundred tickets, with the following six categories of root causes:

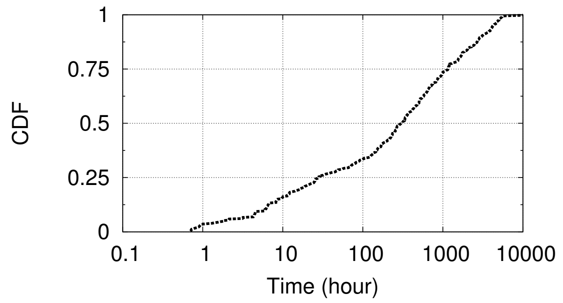
- *Maintenance*: expected or scheduled network actions or changes;
- *Circuit*: connection between two devices (on specific interfaces) is down.
- *Cable*: cable disconnection due to environmental or human artifacts.
- *Hardware*: failures of cards that constitute the chassis system and components that constitute a card.
- *Software*: failures due to software issues.
- *Duplicate*: follow-up failures when the original issue is not resolved.

For each trouble ticket, we track both the ticket report time and the repair finish time. Trouble tickets are triggered by signals from various network monitoring systems matching against known problem signatures, via a series of ticket processing logic, such as pattern matching and event correlation. Thus the ticket report time is often at or after the first occurrence of a symptom of the network fault. Since the ticket generation process is imperfect, it may miss early symptoms and introduce significant delays between the first occurrence of symptoms and the actual generation of a ticket.

VNF Syslog. Syslogs are complex, unstructured, free-form texts generated by the systems to describe a wide range of events [123, 124]. One vPE could have millions of syslog messages per year. Both keywords and relationships among different types of log messages [118, 124, 125, 126] define the key structural patterns of syslogs. We use



(a) Percent of types over time (monthly).



(b) Non-duplicated ticket inter-arrival time.

Figure 3.3: Ticket analysis of aggregated vPEs.

the well-known *Signature tree* [124] approach to transform raw syslogs into a structured representation for convenient relationship modeling.

We also compare our vPE syslogs to those of pPEs (physical Provider Edge routers) with similar number of network tickets. We observe that vPE syslogs have 77% less volume than pPE syslogs, and contain many fewer log messages on physical layer. This confirms our intuition that NFV reduces each vPE's visibility of lower layer events.

3.3.2 Trouble Ticket Analysis

To help understand the predictability of trouble tickets, we focus our analysis on (1) ticket temporal distribution/frequency and (2) similarity of ticket patterns between vPEs.

Temporal Distribution. Figure 3.3(a) shows tickets with different root causes over time. We found that maintenance is the dominant factor, but they are predictable (since they are prescheduled events). Duplicated tickets and circuit tickets are the next two major contributors. Overall, the ticket data is highly skewed. Figure 3.3(b) plots the distribution of inter-arrival time of non-duplicated tickets per vPE. We see that non-duplicated tickets arrive more than 40 minutes apart. 80% of time gaps between consecutive tickets are longer than 10 hours, and 25% of gaps between consecutive tickets are longer than 1000 hours (42 days). Finally, we observe that duplicated tickets often

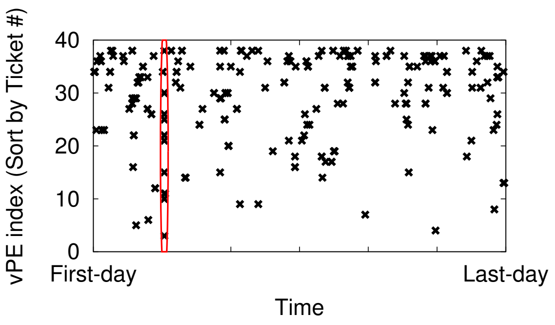


Figure 3.4: Tickets distributed across time and vPEs.

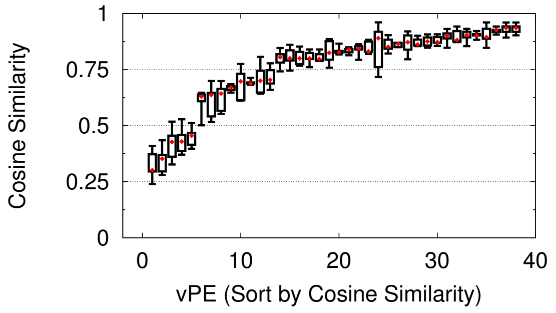


Figure 3.5: Cosine similarity of syslog distribution between all vPEs and individual vPE.

arrive in bursts.

Per vPE Ticket Behaviors. Figure 3.4 shows non-maintenance trouble tickets across vPEs (sorted by their ticket volume per vPE). Each point indicates that the corresponding vPE (y) has ticket on a given time interval (x). Clearly the ticket pattern is non-periodic and vPE-dependent – a few vPEs have far more tickets than others. There is no obvious bias in time or towards any specific vPE. Another observation is that sometimes, multiple vPEs experience network faulty conditions in the same time interval (marked by the vertical bar). A deeper look at the data showed that these tickets are triggered by issues of core routers that led to disruptions at all attached vPEs. However, such cases are very rare, and only contribute to a very small percentage of trouble tickets.

3.3.3 VNF Syslog Analysis

We perform temporal and spatial analysis on VNF syslogs collected at vPEs. To analyze "normal" syslog entries unrelated to network failure events, we prune the log to remove any entries that are within 3 days of a ticket's active period (the period between a ticket's arrival time to when it is marked as resolved).

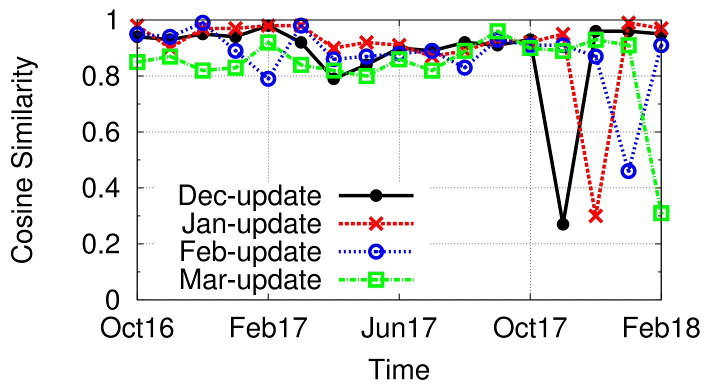


Figure 3.6: Cosine similarity of syslog distribution between this and next month.

Correlation across vPEs. We first ask the question: do vPEs' syslogs display similar behaviors during normal operations (*i.e.*, no failures)? We compute the cosine similarity [127] of syslog distributions for each vPE v, and that of the aggregated syslog over all vPEs V, *i.e.*, $\frac{\Sigma_{i=1}^{n} s(v)_i s(V)_i}{\sqrt{\Sigma_{i=1}^{n} s(v)_i^2}\sqrt{\Sigma_{i=1}^{n} s(V)_i^2}}$, where $s(\cdot)$ denotes the syslog distribution. We use a sliding time window of one month across syslogs, and calculate the normalized frequency distribution.

Figure 3.5 shows the quantile values (0%, 25%, 50%, 75%, 100%) of the cosine similarity across time. Only one third of vPEs have a similar syslog distribution (cosine similarity > 0.8), and there are 5 vPEs that have < 0.5 in cosine similarity. This indicates that syslog patterns vary across vPEs, possibly due to differences in server roles, configurations and traffic. Therefore, we will need per-vPE customized models to detect anomalies on vPE syslogs.

Impact of System Updates. Another key finding is that some vPEs' syslogs had sudden changes between late 2017 and early 2018, triggered by system updates that changed the syslog distribution. We compute the cosine similarity of syslog distributions between consecutive months. Figure 3.6 shows the averaged results. We found that before the system updates, cosine similarity is consistently above 0.8, but drops below

0.4 following a system update. This means that we need to update models of vPE syslogs quickly (using short windows of data), so that they do not become obsolete.

3.4 Predicting Tickets from Syslog Anomalies

In this section, we describe our effort to identify specific (or anomalous) patterns in vPE syslogs that may potentially serve as early detection or warning signatures for (trouble) ticketing conditions.

3.4.1 Methodology

Our empirical analysis in §3.3 identifies three key challenges for predicting trouble tickets via vPE syslogs. *First*, trouble tickets are relatively rare across our vPE syslogs. With such imbalanced data, it is very difficult to train a supervised learning model for fault prediction. *Second*, the volume and complexity of syslog data make it difficult to manually select the feature set necessary to train ML models on log behavior. *Third*, since syslog distributions vary across vPEs and over time, we need to customize machine learning models for each vPE, and re-train them after system updates. Both can lead to large overheads in terms of data collection delay.

To address the first two challenges, we build a Long Short-Term Memory (LSTM) network [121] that *automatically* learns syslog patterns during normal operations (§3.4.2). Instead of supervised training, we take an anomaly detection approach using a baseline model trained using "normal" syslog data. Thus we are unaffected by the rarity of trouble ticket events. Each detected anomaly can potentially serve as an indicator for network faulty conditions. To address the third challenge of data collection latency, we apply both clustering and online learning techniques to reduce the amount of training data required to customize models for individual vPEs (§3.4.3).

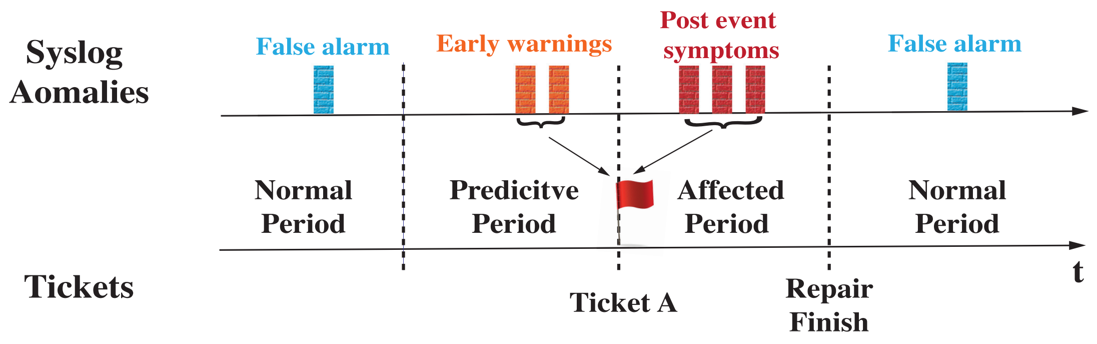


Figure 3.7: Mapping syslog anomalies to trouble tickets.

time window ahead of the ticket generation as the *predictive period*, and the time between ticket report and repair finish as the *affected period*. As shown in Figure 3.7, if an anomaly is detected during the predictive or affected period of a ticket, we associate that anomaly to the ticket. Specifically, an anomaly detected during the prediction period of a ticket is treated as an "early warning signal," and those detected during a ticket's affected period are treated as "post event symptoms." Although there are many reasons why anomalies may occur before the ticket time, some of the early warning signals may be converted into alternative ticket-triggering signatures. Anomalies which are not associated with tickets will be treated as false alarms. We vary the length of the predictive period to see performance changes in Section 3.5.

3.4.2 LSTM-based Anomaly Detection

As a language for communication between users/programs and the system, vPE syslogs display sequential patterns. An accurate model of syslogs must be able to capture those sequential patterns. Thus we consider the Long Short-Term Memory (LSTM) network, which is well-known for its capability of capturing the comprehensive and intricate patterns embedded in sequential data[2]. With sufficient training data, LSTM can automati-

[2]LSTM is a special case of Recurrent Neural Networks (RNN). It is equipped with explicit memory cells that have the ability to remember long-term dependencies over sequences. [121] provides a detailed

cally learn normal patterns of syslogs, and flag deviations from the norm as anomalies. In fact, LSTMs have demonstrated great success in detecting a wide range of anomalies, such as server faults in distributed systems or anomalies in sentiment analysis [128, 126, 125].

Unlike traditional linear classifiers, our approach does not rely on feature engineering. For the input of LSTM, we use each individual log m_i, which captures system events for a specific interval $([t_i, t_{i-1}))$ (m_i appears at t_i). Instead of using just raw log entries, we apply the aforementioned signature tree approach [124] to extract and categorize a specific template (or signature) from the raw data, marked by a tuple of $(m_i, t_i - t_{i-1})$, $m_i \in S$, where S is the template collection. Given k syslog tuples, we train our LSTM model to predict m_{k+1}. This is a multi-class classification problem where the output is a probability distribution over the template set S.

Model Training. We train the LSTM network using syslogs produced during "ticket-free" network operations. As mentioned in Section 3.3.3, we prune syslog entries that occur within a 3-day buffer around the active window of actual tickets. We also experimented with larger window sizes but did not observe notable differences.

Detecting Anomalies. Using a trained LSTM model, we detect an anomaly as follows. To determine whether an incoming syslog m_{k+1} is normal or abnormal, we plugin the previously observed k syslogs into the model and derive the probability distribution of prediction of the $(k+1)$th log. If m_{k+1} is normal, then the corresponding log-likelihood value should be high (above a threshold), and abnormal if not. By changing the threshold value, we can derive a precision-recall curve (PRC), which is the most widely used measure to evaluate anomaly detection systems [129].

Learning Minority Syslog Patterns. While LSTMs are designed to automatically learn patterns of normal syslog entries, minority patterns are generally hard to learn

tutorial of LSTM.

given their rare appearances in the training data. The result is a high false alarm rate. We address this by over-sampling the minority (normal) patterns [130]. Specifically, we use month i's syslog to train a LSTM model that will be used to detect anomalies during month $(i + 1)$. We apply the LSTM model training in multiple rounds, using month i's normal syslog as training data. After each round of training, we test the model using the original training data and identify normal syslog patterns that are misclassified as anomalies. We then over-sample these patterns and randomly sample all other patterns, and use the resulting data to adjust the model weights. The process exits when the false positive rate cannot be further improved.

3.4.3 Customization and Adaptation

Since the syslog distribution varies across vPEs, a general LSTM model will likely achieve suboptimal accuracy. The ideal solution is to build a customized model per vPE, but the resulting training overhead and data collection latency are unacceptable. We address this tradeoff between model accuracy and data collection latency using vPE grouping [131]. We apply K-means [132] to group vPEs and choose the number of groups K based on modularity. vPEs in the same cluster show similar patterns in syslog distributions, and their training data will be aggregated together to build a unified model for the group. For our dataset, we produced 4 vPE clusters, which led to 4 LSTM models.

We also reduce the latency of training data collection using online (or incremental) learning. Specifically, each month we perform a round of incremental training by updating the model weights using the newly arrived syslog entries. Since the syslog distribution is relatively stable, we do not observe significant changes in model weights.

The exception is that between late 2017 and early 2018, the vPE network had a system upgrade, and some vPEs' syslog distributions were significantly modified. As a result, the number of false alarms increased by a factor of 14, indicating that the model is out-of-date and required updating. The naive solution is to retrain the entire model, but rebuilding a reasonable training dataset takes more than 3 months. We want a solution that can retrain the models in a much shorter time window.

To address this challenge, we consider transfer learning [133], where a pretrained neural network model (*i.e.* a "teacher model" that was trained before the system update) is adapted using limited training data to a student model that can respond to new syslog behavior. Specifically, we build the student model by first copying the teacher model, then training the student model using new syslog data to fine tune the top layers of the model. For our cases, one week of new training data is sufficient to quickly update the model after a major software update.

3.5 Evaluation

In this section, we evaluate our LSTM-based anomaly detection system, and the feasibility of using vPE syslog anomalies as (early) warning signatures of network trouble tickets.

3.5.1 Experimental Setup

We implemented our anomaly detection system using Keras [134] with Tensorflow [135] as the backend. For model optimization, we varied model parameters to minimize the categorical cross entropy [136], but found that model performance is generally insensitive to parameter choices. Our final LSTM model consists of 2 LSTM layers and 1 dense layer.

Estimating Ground Truth of Syslog Anomalies. Evaluation of our anomaly detection system requires ground truth of syslog anomalies, which we approximate using

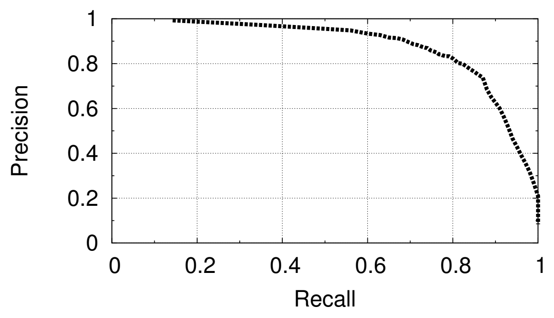


Figure 3.8: PRC for 1-day predictive period.

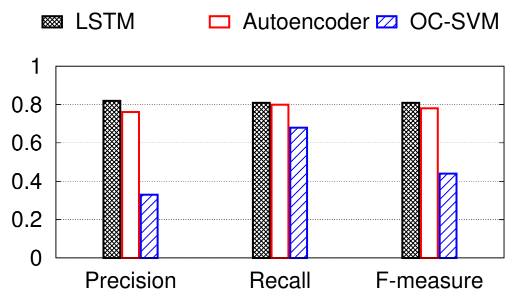


Figure 3.9: Anomaly detection performance of different approaches.

trouble tickets. For each trouble ticket, we define a time window before its generation time as the *predictive period*, and the time window after its generation till the reported ticket repair time (ticket duration) as the *affected period*. As shown by Figure 3.7, if any syslog anomaly falls into the predictive period or the affected period of a ticket, we will treat it as a true anomaly. So one ticket can possibly have multiple (early) signatures. On the other hand, any anomaly outside of these periods is treated as a false positive. We tried multiple values of predictive periods, from 1 hour to 2 days, and found that the detection performance converges at 1 day.

Another interesting observation is that after matching syslog anomalies with non-duplicated tickets, each ticket is associated with at least two anomalies (in the predictive period). These anomalies are close to each other, less than 1 minute apart on average. Thus we configure the detection system to report a warning signature for network trouble tickets upon detecting a small cluster of two or more anomalies.

Training and Testing. We use syslog data from the first month of the 18-month data for initial model training. At the end of each month, we update the LSTM model using fresh data from the previous month, and test the updated model using data from the subsequent month. Both initial model training and monthly model updates complete in less than one hour.

3.5.2 Accuracy of Anomaly Detection

Precision, Recall, F-Measure. We start with three standard metrics on anomaly detection [137]. Precision shows the percentage of true anomalies among all anomalies detected; Recall measures the percentage of anomalies in the data set (tickets as the ground-truth) being detected; and F-measure is the harmonic mean of the two.

Figure 3.8 plots the Precision-Recall Curve (PRC) produced by adjusting the aforementioned threshold in LSTM log probability (§3.4.2). Our final operating point is the one that maximizes the F-measure, with precision at 0.8 and recall at 0.81. In this case, our system can effectively identify anomalies while achieving low false positives at 0.6 per day for all vPEs.

Comparison to Existing Methods. We consider two existing methods on anomaly detection:

- *Autoencoder* [138] is a feed-forward multi-layer neural network in which the desired output is the input itself. After training the auto-encoder with normal data, the reconstruction error can be used as an anomaly indicator. We use the TF-IDF (term-frequency, inverse document frequency) Features [139] as the input to Autoencoder.

- *One-Class SVM* [140] uses shallow learning to build a model of the normal syslog training data, which requires feature engineering (mapping the data into a high dimensional feature space via a kernel). If a new syslog entry deviates significantly from the model, it is marked as anomaly.

For a fair comparison, we applied the same customization and adaption mechanisms (§3.4.3) on all three approaches.

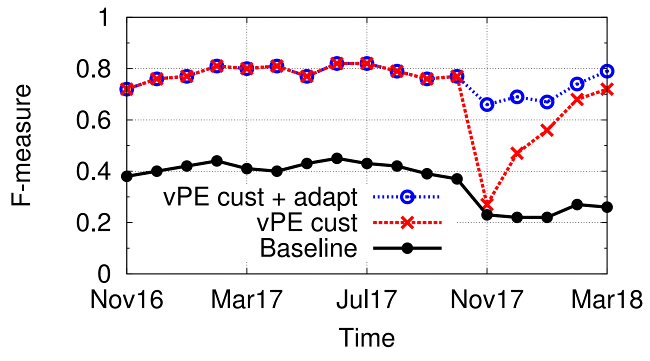


Figure 3.10: Effectiveness of different components.

Figure 3.9 shows the performance of the three approaches. The two deep learning approaches (LSTM, Autoencoder) largely outperform the traditional classification approach (one-class SVM), because feature engineering is highly challenging given the volume and complexity of the vPE syslogs. LSTM slightly outperforms Autoencoder (a precision of 0.82 vs. 0.77), by capturing sequential patterns of the syslogs.

Gains of Customization and Adaptation. We use microbenchmarks to understand the contribution of model customization (a single model for all vPEs vs. customized models per vPE) and fast model adaptation (following a system update). Figure 3.10 plots the model F-measure across the 18 month period. Model customization produces significant improvement in model F-measure and precision (results not shown due to space limits). Our model adaptation component allows the system to quickly recover from disruption caused by software updates using just 1-week of training data. Using training data longer than 1 week does not produce significant improvements.

Reducing Training Overhead. Our design uses both vPE clustering and transfer learning to reduce the amount of syslog training data (for constructing and adapting the LSTM model). We evaluate their effectiveness by comparing each to their corresponding baselines. Using vPE clustering, we are able to reduce the amount of (initial) training data from 3 months to 1 month. Using transfer learning, we reduce the recover time (from

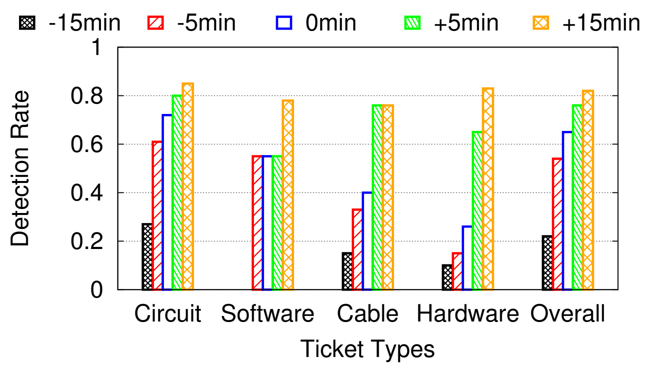


Figure 3.11: Anomaly detection for different types of tickets: X time after ticket generation.

software updates) from 3 months down to 1 week. This means we can build and maintain a high-quality prediction model without incurring expensive delays for collecting training data.

3.5.3 Trouble Ticket-based Evaluation

We use trouble tickets as approximate ground truth to evaluate how effectively our method can discover anomalous syslog conditions. Figure 3.11 shows the probability of detecting any anomaly related to a ticket (at least 15 minutes prior to the ticket arrival, at least 5 minutes prior, 0 minute prior, until 5 minutes after, and until 15 minutes after) for each individual (non-duplicated) ticket type, and across all the tickets.

We seek to answer the following questions:

Q1: What types of network trouble tickets show early signs in VNF syslogs?

Answer: We discover VNF syslogs appear before multiple trouble ticket types (e.g., Circuit, Software, Cable and Hardware). Syslogs related to circuit failure tickets have the highest probability of occurrence before the ticket generation (74%), followed by Software (55%), Cable (40%) and Hardware (28%). This indicates that despite reduced visibility into lower faults caused by virtualization, VNF syslogs do capture anomalous conditions related to network trouble tickets.

Q2: For failures that do not display syslog anomalies before ticket generation, will any of their anomalies show up to the syslog shortly?

Answer: Yes, for majority of tickets (80%), vPE syslogs will display anomalous patterns within 15 minutes after the ticket generation. This means that patterns of failures become visible at the NFV layer after a small delay, which can be leveraged by NFV for trouble ticket analysis, diagnosis and management.

Q3: How early do we observe syslog anomalous conditions compared to ticket generations?

Answer: The majority of detected syslog anomalies are 5 minutes ahead of the ticket generation. For Circuit, 36% of syslog anomalies are 15 minutes ahead, and the ratio is even higher for Cable (39%) and Hardware (38%) categories.

Although more in-depth investigation is required, these results indicate the possibility that operators may be able to leverage these syslog anomalies to either improve their ticketing process, or identify predictive or early conditions indicative of network failures.

Q4: Can a single or group of anomalies serve as warning signatures for a group of near-term trouble tickets?

Answer: This is related to the question of whether a single syslog anomaly (or a cluster of syslog anomalies) can be associated with multiple trouble tickets. Based on our current dataset, this has never happened, mostly because the tickets are rare and well-separated. We plan to confirm this finding using larger-scale studies in the future.

Operational Findings. The anomalies identified by our model can be categorized into four scenarios. First, the detected conditions are likely true predictive signals for near-term network problems. For example, we identified a condition that involves a management daemon error message about some peer session connection failures with a particular controller (*"invalid response from peer chassis-control"*). When an anomaly with this condition was observed, it was typically followed some time later by a trouble ticket. We need to investigate this apparently predictive signature further to understand the underlying vPE behaviour. Second, the detected conditions can be analyzed and turned into early detection signatures on faulty conditions. For example, we found that a storm of protocol session flaps (*"BGP UNUSABLE ASPATH: bgp reject path"*) across multiple peers within a short time interval can be turned into a quick detection signature (with minimum false positives). This anomaly detection outperforms existing service level monitors, which normally have a longer detection delay. Third, the detected conditions could be part of the events that triggered the trouble tickets. This may be due to event response procedures in existing ticketing process flows, such as intentional delays added to suppress transient issues. Our findings may help operations to further optimize such ticketing process flows. Fourth, the detected conditions are coincidental to the ticket (*i.e.,* involving unrelated syslog anomalies). This scenario is relatively rare and should be carefully managed, *e.g.,* through adding suppression rules in ticket processing flows. In future work, we will further categorize the detected conditions into these four scenarios.

3.6 Related Work

Reliability and Fault Management in NFV. [141, 142] addressed the necessity and challenges of reliability, resiliency and fault management in NFV, showing that one of the key challenges is the cooperation and latency between layers. [143] studied the correlation among network resource alarms and produced rules for root cause analysis. [144, 145] leveraged Self-Organizing Map (SOM)-based clustering to identify different types of network failures based on SNMP measurements, but requires sufficient samples of each failure type in advance. [146] collected metrics from both hypervisor and VM layers, and applied Random Forest to classify VNF behaviors. All of these evaluated small-scale, self-defined network failures.

Trouble Prediction/Detection in Networking. While existing works [131, 147] achieve trouble detection based on Key Performance Indicators (KPIs), such as CPU utilization and packet loss, our work focuses on VNF syslogs. The majority of existing works apply supervised trouble prediction/detection, by building binary classifiers that are trained with both normal and abnormal events. [148, 149, 150] applied simple failure prediction methods based on characteristics of failure events, and developed Hidden Markov Model (HMM) and shallow machine learning approaches for network failure prediction. To capture sequential patterns in the monitoring data, [118] designed sequential features and applied Random Forest to learn omen and non-omen patterns for switch hardware failures in data centers. [139] applied LSTM to detect a single type of failure for server cluster down. The key challenge faced by the above supervised methods is that they require sufficient anomalous data to train the model, which takes a significant amount of time to collect, *e.g.* multiple years according to the above studies.

To reduce data collection latency, several works resorted to unsupervised approaches. [123] extracted features on state variables and identifiers, and applied PCA to perform anomaly detection. [126, 125] applied LSTM on network intrusion detection for Linux system calls and OpenStack experiments on CloudLab. While we also take an unsupervised learning approach, our work differs from existing works by focusing on predictive analysis of failures in NFV systems.

3.7 Conclusion

We use system log and network trouble tickets in a real-world deployment to study the problem of failure prediction in NFV networks. We propose a new method to detect anomalies from NFV syslogs that can potentially be used as early indicator of network issues that would typically result in trouble tickets. We validate our methodology using a sample dataset collected over 18-months on virtualized provider edge (vPE) routers in a production NFV environment. We observed that our LSTM-based anomaly detection system discovers syslog anomalous conditions that often occur before the trouble tickets. We believe our methodology can help the network operations teams to either (a) identify predictive or early warning signals, or (b) improve upon the current ticketing process that will enable timely response to NFV failures.

Chapter 4

Scaling DNN Models for Spectrum Anomaly Detection

Spectrum management in cellular networks is a challenging task that will only increase in difficulty as complexity grows in hardware, configurations, and new access technology (*e.g.* LTE for IoT devices). Wireless providers need robust and flexible tools to monitor and detect faults and misbehavior in physical spectrum usage, and to deploy them at scale. In this chapter, we explore the design of such a system by building deep neural network (DNN) models[1] to capture spectrum usage patterns and use them as baselines to detect spectrum usage anomalies resulting from faults and misuse. Using detailed LTE spectrum measurements, we show that the key challenge facing this design is model scalability, *i.e.* how to train and deploy DNN models at a large number of static and mobile observers located throughout the network. We address this challenge by building context-agnostic models for spectrum usage and applying transfer learning to minimize

[1]Z. Li, et al, Scaling Deep Learning Models for Spectrum Anomaly Detection, Proceedings of 20th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Catania, Italy, July 2019. Our proposed spectrum model, code, and test dataset are available at https://github.com/0x10cxR1/spectrum_anomaly_detection/.

training time and dataset constraints. The end result is a practical DNN model that can be easily deployed on both mobile and static observers, enabling timely detection of spectrum anomalies across LTE networks.

4.1 Introduction

Cellular providers spend billions of dollars acquiring radio spectrum for network capacity and coverage. Yet spectrum management, specifically detection of faults from spectrum interference, remains a costly and ad hoc process, often involving manual diagnosis following customer complaints and operational failure logs. What makes detection hard is that interference can come from a variety of complex sources at any physical location, ranging from intentional spectrum misuse and misconfigured transmitters to RF leakage from cable plants and connectors. For example, interference from a misconfigured amplifier led to persistent quality-of-service issues for a tier 1 service provider [151].

These problems will grow in severity and scale in the near future. Advances in both reconfigurable hardware and spectrum usage policies make it easy to misuse spectrum without authorization. There is already evidence of these misuse attacks in China, where growth in unauthorized transmissions has prompted new initiatives to outlaw spectrum misuse [152]. Furthermore, cellular interfaces for IoT are coming, optimized for the network and energy needs of IoT devices. Adoption of these interfaces has the side effect of increasing security risks for LTE and nearby spectrum bands. A compromised device working on behalf of an attacker can perform jamming or denial of service attacks on cellular bands.

Clearly, cellular networks need robust and flexible tools to detect faults and misbehavior in spectrum usage, which we hereby refer to as *spectrum anomalies*. Despite open calls for automated management tools by the 3GPP standards body, current pro-

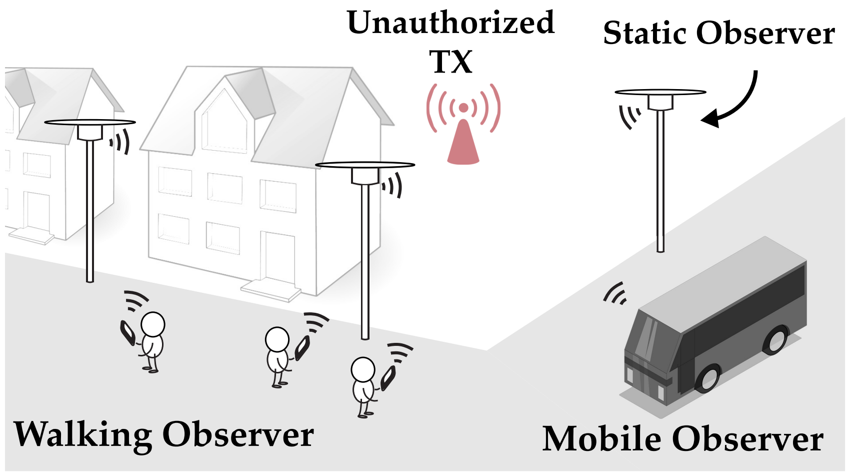


Figure 4.1: Spectrum anomaly detection by multiple observers.

posals are still limited to simplistic fail-stop fault models, and only on faults *within* the LTE infrastructure [153, 18]. Cellular carriers often evaluate physical spectrum usage by wardriving with specialized devices, and these activities are severely limited by high human and equipment costs [154].

Instead, we believe that cellular networks require *general* solutions capable of detecting a range of radio spectrum anomalies, from *transmissions at unexpected power levels*, to *interference from misconfigured devices* and *unauthorized transmitters*. Anomalies can appear anywhere in the physical network, and their detection requires a large-scale, distributed spectrum monitoring system.

In this paper, we explore the design of a general, scalable system for detecting spectrum anomalies in wide-area LTE networks. As shown in Figure 4.1, the system consists of two components: (1) a scalable, distributed spectrum monitoring system that measures physical spectrum usage using both static and mobile observers[2] distributed across the network, and (2) a general anomaly detection system that builds deep neural network

[2]Spectrum monitoring requires both static and mobile observers to enforce coverage and scale. We assume that these observers are recruited by the carriers to perform spectrum monitoring and anomaly detection, and are well-behaved. This simplification allows us to focus on the problem of scaling DNN models for anomaly detection.

(DNN) models using these measurements, and runs them at each observer as baselines to detect spectrum usage anomalies. Our work builds on multiple prior efforts: some of which examined the feasibility of distributed spectrum monitoring using commodity devices [155, 156], while others validated the benefits of DNN-based spectrum anomaly detection using a single static observer [157, 158].

Why DNNs? Wide-area spectrum measurements collected by observers are highly complex, thanks to unpredictable signal propagation, frequent link adaptation, and traffic dynamics. Traditional models are unable to capture such complexity. DNN models, on the other hand, are known for automatically capturing complex patterns in the target data. Recent works have demonstrated the advantages of using DNNs to model spectrum usage [157, 158].

Despite significant progress, a large gap remains between current proposals and a feasible system for cellular networks. Since spectrum measurements generally depend on the context of the observer, *i.e.* time, location, and mobility status, each observer should ideally run a model tailored to the current context. But this renders our system impractical, given the amount of training overhead and run-time complexity it requires, *i.e.* it is impractical to assume that the system must build models for each physical location, and that each observer must change its model whenever it moves. A practical alternative is to explore a context-agnostic model for all observers, and whether such models can be trained, deployed and validated. *But will the elimination of "context-awareness" from DNN model designs degrade the accuracy of spectrum anomaly detection*?

We answer these questions through an empirical study on LTE networks, using detailed spectrum measurements across multiple LTE bands and cells. Our efforts lead to three key findings:

- Within each LTE cell, it is feasible to build a single, context-agnostic DNN model that accurately models normal spectrum usage pattern for the task of anomaly detection. Our DNN model does not use supervised learning to classify an event as normal or anomalous. Instead, we train a long-short term memory (LSTM) model on sequences of spectrum measurements. It recognizes events as anomalies when they deviate significantly from events expected or predicted by the model. Our model runs on both mobile and static observers to detect spectrum anomalies on the fly without any modification, putting a hard limit on the training overhead and run-time complexity. Deep autoencoder, another DNN model, can be designed to offer the same properties.

- Across LTE cells, the DNN model trained for a given LTE cell is not directly reusable at the other cells, but can be used to quickly train their models through *transfer learning*. Only a small amount of local spectrum measurements at the target cell is required. Our results show using transfer learning instead of training from scratch reduces required training data by a factor of 288.

- Since different LTE bands (frequency carrier, downlink or uplink) display different spectrum patterns, they require different DNN models customized for that band. The same transfer learning method can be applied to quickly train the model for a frequency band using existing models for other bands as a starting point.

Together, these findings demonstrate the feasibility of deploying a practical model for LTE spectrum anomaly detection on top of the distributed spectrum monitoring system. Specifically, the system first trains a general DNN model for normal spectrum usage, *i.e.* the teacher model, using past spectrum measurements from trusted observers. It then distributes this teacher model to each individual LTE cell's basestation, who uses a small amount of local spectrum measurements (contributed by trusted observers in the cell) to quickly calibrate the model, and distributes a unified, context-agnostic model to all the observers in the cell.

The above design has two key features. First, the spectrum DNN model is context-agnostic and can be easily deployed on a wide range of spectrum observers, static and mobile, and adapted using a minimal amount of local spectrum measurements. Each observer does not need to store a large number of models for each context, or switch to a new model whenever it moves. Instead, it runs the same DNN model regardless of its context, and only needs to switch to a new model when moving into a different cell. Second, the anomaly detection is general in that it avoids cellular-specific knowledge and can detect any events that affect spectrum usage.

4.2 Preliminary

To provide context for our later discussions, we present in this section the spectrum measurement dataset used in our empirical study, and our initial analysis on patterns in today's LTE spectrum usage. We also present existing models for spectrum anomaly detection, and evaluate their performance using our spectrum measurements in the presence of spectrum anomalies.

4.2.1 Spectrum Anomalies

The goal of our anomaly detection system is to detect a wide range of issues ranging from hardware and software failures, to misconfigurations and intentional misuse of spectrum. While it may be intractable to predict all possible sources of failures or performance issues in a cellular network, the observable impact of these faults on spectrum usage is often easier to categorize. Here, we summarize several of the most common observable spectrum anomalies, and where possible, we list potential causes.

Unauthorized Transmitter. These are cases of observable active transmitters that do not have a license to transmit. These errors can be caused by a number of faults, including accidental misconfiguration leading to a transmitter operating outside of its normal frequency range [159], intentional unlicensed transmissions avoiding costs of a spectrum license, or transmissions with the intent to disrupt existing services (signal jamming attacks). These transmissions can produce a partial or full overlap in the frequency domain with existing transmitters, using either the same or different signal modulation as the victim.

Misconfigured Transmitter (in power). These are cases of transmitters licensed to transmit on its current frequency, *e.g.* LTE basestations. But due to either misconfigurations or hardware failures, they are not operating stably at their normal transmit power, *e.g.* the aforementioned incident of misconfigured amplifier [151].

Misconfigured Transmitter (in frequency). Similar to the above, the radio hardware fails to operate on some of its operating frequencies. In an extreme case, the transmitter stops to operate on all its frequencies and shuts down completely (TX fault).

4.2.2 Analysis of LTE Spectrum Usage

Our Dataset. We performed signal measurements on three major LTE carriers in the US, including three downlink (DL) bands of AT&T (880MHz), T-Mobile (729MHz), and Verizon (749MHz), and one uplink (UL) band of AT&T (830 MHz). We used USRP N210 devices to capture 5 MHz spectrum within each LTE band.

While prior works on spectrum misuse detection [160, 161, 162, 157] only considered static observers, we performed measurements on LTE spectrum usage using both static and mobile observers (walking, driving). Our measurements were performed at two areas: a large university campus and an urban downtown area, separated by a distance

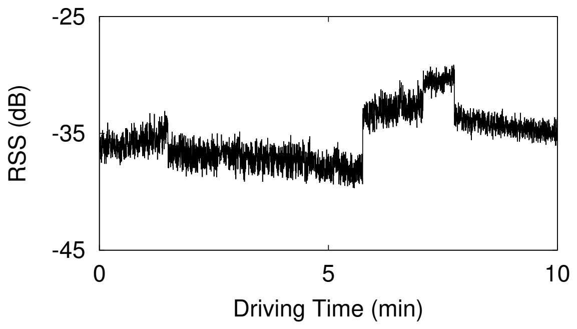
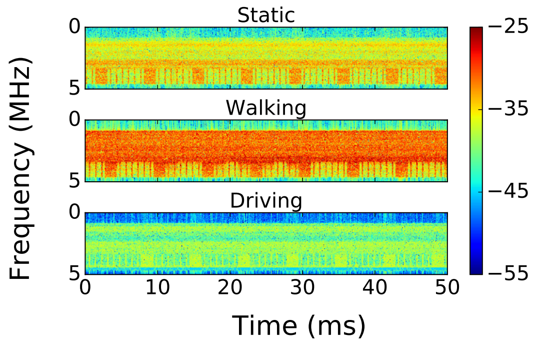
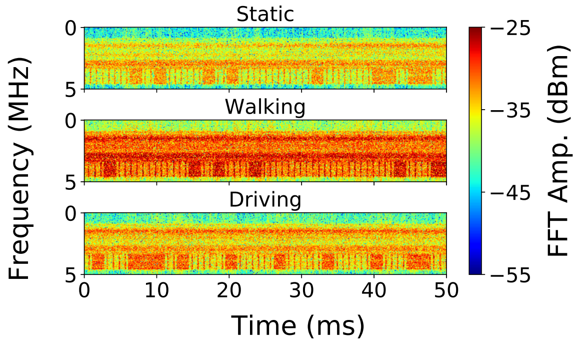


Figure 4.2: RSS varies largely over a 10-minute monitoring window, for a mobile observer.



(a) Initial spectrograms.



(b) Spectrograms after 10 minutes.

Figure 4.3: Spectrograms captured by spectrum observers under different contexts, based on the measurements on the 880MHz downlink band.

of 8 miles. For each area, we verified that the observers were in the same LTE cell during the measurement period and the measurement range is within 1 mile.

Our measurements were performed between January and March 2018, and repeated in June 2018 to examine potential temporal variations. Specifically, we set up three static observers (well separated) in the university campus and collected measurements continuously for 7 days, and two static, well-separated observers in the downtown area for 3 days of continuous measurements. Walking and driving experiments were done for 45 min per day for 8 days. In total, the dataset contains more than 20 TB of signal data, where 32% of the data were collected at night.

Spectrum Usage $\neq$ RSS. Many prior works [160, 161, 162, 163, 164] have used measured received signal strength (RSS) as the base for spectrum anomaly detection, where an anomaly occurs if the current RSS deviates from a pre-defined range. Our measurement shows that RSS is not a viable base for mobile observers since it changes significantly and unpredictably over time. Figure 4.2 shows a random segment of RSS collected by a mobile observer over 10 minutes. Here the sudden rise of RSS values can be the result of multipath fading or interference from an unauthorized transmitter in proximity, which are indistinguishable using RSS data.

Time-Frequency Patterns of Spectrum Usage. Instead, we chose to analyze spectrum usage using the time-frequency spectrogram of the received signal. Spectrograms capture fine-grained signal amplitude over time at sub-frequencies, and are widely used for spectrum analysis. In absence of any anomaly, Figure 4.3 plots a spectrogram segment of $50ms$ at three observers (static, walking, driving) and another $50ms$ segment at each of the same observers about 10 minutes later. Despite the large difference in signal amplitude across users and time, we can observe visible temporal patterns from all six segments, in the form of bursts of high-power transmission along the time dimension.

We studied these patterns in detail and arrived at two key observation. *First*, the pattern is complex, especially in the temporal domain. Periodicity analysis shows that signal fluctuation peaks reside at 1200Hz, 160Hz and 60Hz, indicating that the key periodic pattern occurs every $0.21ms$, $1.6ms$, and $6ms$. More frequencies of transmission bursts exist in addition to these main peaks, indicating more fine-grained temporal patterns beneath the obvious bursty patterns we observed. *Second*, the *short-term* patterns of the spectrum usage share some general shape. Carefully formed, they could serve as reliable "fingerprints" of normal spectrum usage.

Spectrum Patterns across Time, Cells and Bands. We also visually compared the spectrum usage patterns observed across time, LTE cells, and LTE bands. The short-term usage patterns are fairly consistent over time (by comparing observations in January-March, and June), differ slightly across cells (campus vs. downtown), but show more visible differences across LTE bands. We also performed periodicity analysis to confirm these observations.

4.2.3 Models for Spectrum Anomaly Detection

The above analysis suggests that it is feasible to build general spectrum anomaly detection by modeling the time-frequency patterns of normal LTE spectrum usage. The hypothesis is that the presence of a spectrum anomaly will produce visible changes to the patterns extracted from the measured signals, which trigger the detection of the anomaly. This type of anomaly detection prioritizes generality: training/building the model using normal spectrum usage *without* requiring any knowledge or labeling on anomaly instances.

There are multiple existing approaches of modeling spectrum usage patterns from the spectrogram, including:

- *One-class SVM classifier* creates a model of the normal training data by mapping it into a high dimensional feature space via a kernel, and iteratively finds the maximal margin hyperplane which best separates the training data from the origin [162]. New data significantly different from the model will be marked as anomaly. This approach requires feature engineering, and we use the FFT decomposition of the signal RSS over 256ms as the input features.

- *Kalman Filter* [165] is widely used for estimating the state of the system given noisy measurements. An anomaly is detected if the predicted state deviates significantly from the observed one. It is well-suited for detecting sudden state changes via continuous observations.

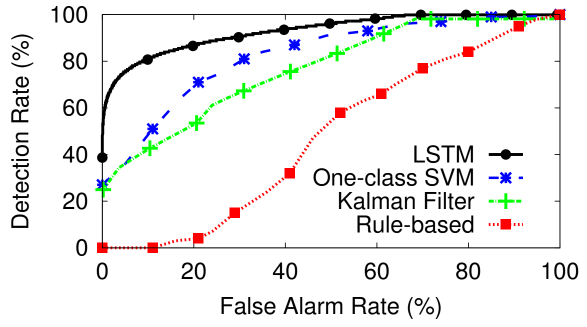


Figure 4.4: Anomaly detection performance of non-DNN (one-class SVM, Kalman filter, Rule-based) and DNN (LSTM) models, based on measurements at the DL 880MHz band.

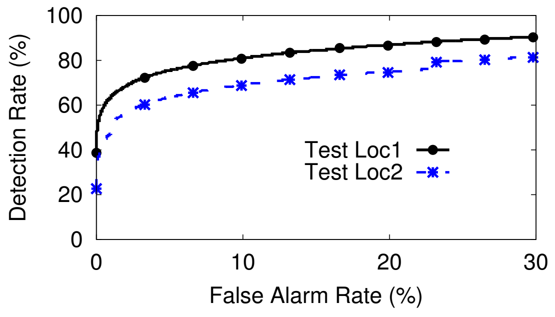
- *Rule-based classifier* detects anomaly when the observed RSS violates a certain threshold [160, 161, 163, 166].

- *Neural Network-based anomaly detector* provides good models without the necessity of feature engineering. Autoencoders [158], and Recurrent neural networks (RNNs), more specifically LSTM (long-short-term memory) models [157] are recently proposed in spectrum anomaly detection.

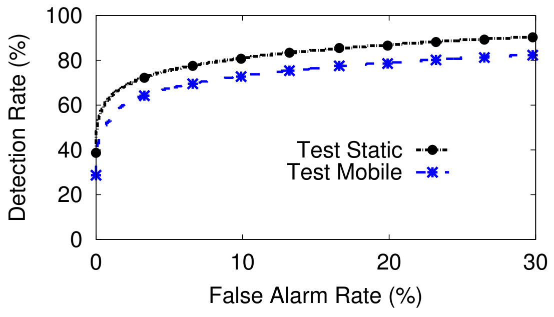
Yet existing works only considered static observers.

We implemented and evaluated these approaches using our LTE measurements. A small portion of our measurements were conducted when anomalies were present. More details on these anomalies are described later in §4.6.1. For all the experiments, the observers were placed within $50m$ of the misuse transmitter.

Evaluation at Static Observers. For all the approaches (Kalman filter, one-class SVM, LSTM, deep autoencoder), we used as the model input the signal spectrogram over $256ms$ (we have tested other segment lengths between $32ms$ and $256ms$ and found that they do not change the conclusion). We trained the models using past spectrum measurements in absence of anomalies at each static observer. We also included a RSS-based method that uses a threshold to detect the presence of anomaly (Rule-based). The



(a) Test across locations.



(b) Test cross context.

Figure 4.5: Anomaly detection performance when (re)using a LSTM model customized for a static observer at location 1. (a) running the model at location 1 and location 2. (b) running the model at location 1 and a mobile observer near location 1 (all based on measurements at the DL 880MHz band).

performance of the LSTM and deep autoencoder models are similar so we only included the LSTM result for brevity.

Figure 4.4 plots the results in terms of anomaly detection rate vs. false alarm rate. The results are similar across the four LTE bands so we only show the result in the 880MHz DL band for brevity. We see that the DNN model (LSTM in this case) largely outperforms the three non-DNN alternatives. This finding aligns with that of recent works [157, 158].

The reason behind the above result is that LTE spectrum patterns are complex due to the compound effect of traffic dynamics, RF propagation, and link adaptation. Additional complexity comes from possible correlations between feature dimensions. All of these make it difficult for traditional methods (*e.g.*, one-class SVM) to model the spectrum usage. Manual identification of good features that capture key spectrum patterns requires deep understanding of the data and much heavier efforts on feature engineering. And the complexity exacerbates when building the models for mobile observers.

4.3 Scaling Spectrum DNN Models

Our empirical analysis validates the observation of prior works [157, 158], where each static observer individually trains DNN models to detect spectrum anomalies. But *can such a context-specific model be deployed on a large-scale distributed monitoring system, where the spectrum observers are distributed across a wide area, and can be mobile or static*? Next, we answer this question empirically, testing whether models trained by a given static observer can be "reused" by another static observer at a different location and another mobile observer. Again we observe a consistent trend across all four LTE bands, and show the result for the 880MHz band for brevity.

Test I: Reusing Models across Locations. Using our LTE measurements at three static observers (in the same LTE cell), we apply the same approach of [157, 158] to train, for each observer, the corresponding DNN models (LSTM and deep autoencoder). We then run the models customized for one observer at the other two observers both with and without the presence of spectrum anomalies. We considered a range of spectrum anomalies in the form of unauthorized transmissions used in §4.2.3.

Our results show that when reusing a spectrum LSTM model at a different location, the model is less accurate in capturing normal spectrum patterns. Thus the anomaly detection rate drops considerably (Figure 4.5(a)). The same applies to the autoencoder model.

Test II: Reusing Models at Mobile Observers. We also experimented with "reusing" models trained for a static observer at a mobile observer (walking at 3mph). Both observers were in close proximity (to reduce the impact of location change). Results in Figure 4.5(b) show a similar trend of performance degradation.

Our Focus: Scaling the DNN Models. Together, these experiments suggest that since wireless measurements depend on the context of the observer, *i.e.* time, location, and mobility status, ideally each observer should run a DNN model tailored to the current context. Unfortunately, this is impractical under our targeted scenario because the system must build models for each physical location in the network and user context, and each observer must change its DNN model whenever it moves. Such requirement leads to significant training overhead and run-time complexity.

This motivates us to explore a practical alternative: building a *unified* model for all the observers, with the goal of prioritizing scale and ease of deployment, minimizing training overhead, and maintaining reasonable accuracy. In the following sections, we tackle this new problem in two steps: first designing a single context-agnostic DNN model for anomaly detection in a single LTE cell (§4.4), then extending the single cell model to train models for many other LTE cells and bands using transfer learning (§4.5).

4.4 A Single Model per Cell

In this section, we focus on designing a single DNN model for a single LTE cell, which will be deployed on all observers in the cell without any modification. Our hypothesis is that within a cell, the normal downlink spectrum usage seen by each observer comes from the same basestation, thus we could train the DNN model to capture a *unified* form of spectrum pattern that is context-agnostic, *i.e.* does not depend on mobility pattern and precise location within the cell. For uplink, each observer sees aggregated transmissions from many LTE users, and the normal spectrum usage could also display context-agnostic patterns. Thus our goal is to design models to automatically discover these context-agnostic patterns, and to validate whether they are sufficient for anomaly detection.

Our study considers two DNN models, LSTM and deep autoencoder. Both are known

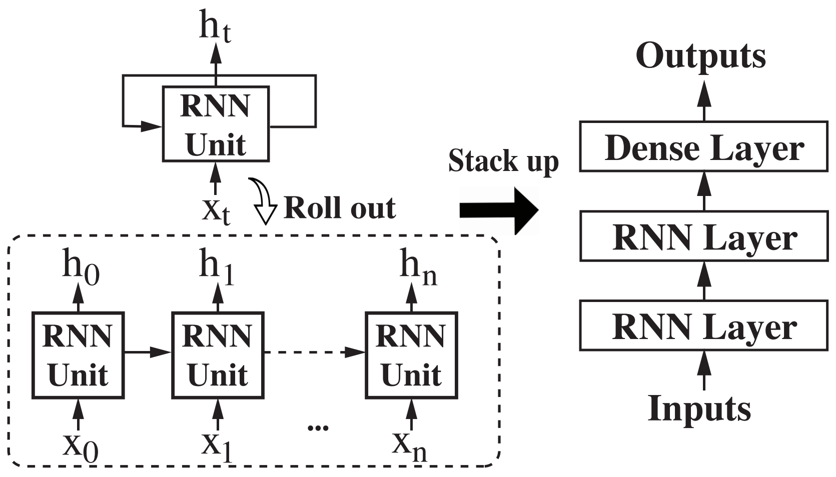


Figure 4.6: RNN unit and stacked RNN network.

for capturing complex, temporal patterns in the target data that can be difficult to detect with simpler models [167, 157, 158]. In the following, we start with a brief introduction of the two models, and then describe the steps taken to build and train a context-agnostic version of these models using our spectrum data. We evaluate the models at both static and mobile observers, in terms of how they predict future spectrum usage. Later in §4.6 we evaluate the corresponding anomaly detection systems.

4.4.1 Background: LSTM and Deep Autoencoder

Recurrent neural networks (RNNs) are artificial neural networks with loops, allowing information to persist [168]. As shown in the left portion of Figure 4.8, its consists of multiple copies of the same network, each passing a message to a successor. Long Short Term Memory networks (LSTMs) are a special type of RNNs, well-known for its capability of capturing the comprehensive and intricate patterns embedded in sequential data. This is achieved by maintaining an internal state in each RNN unit. A practical LSTM model in general consists of multiple stacked layers, forming a more complex architecture similar to feed-forward neural networks [167]. This allows LSTM to learn

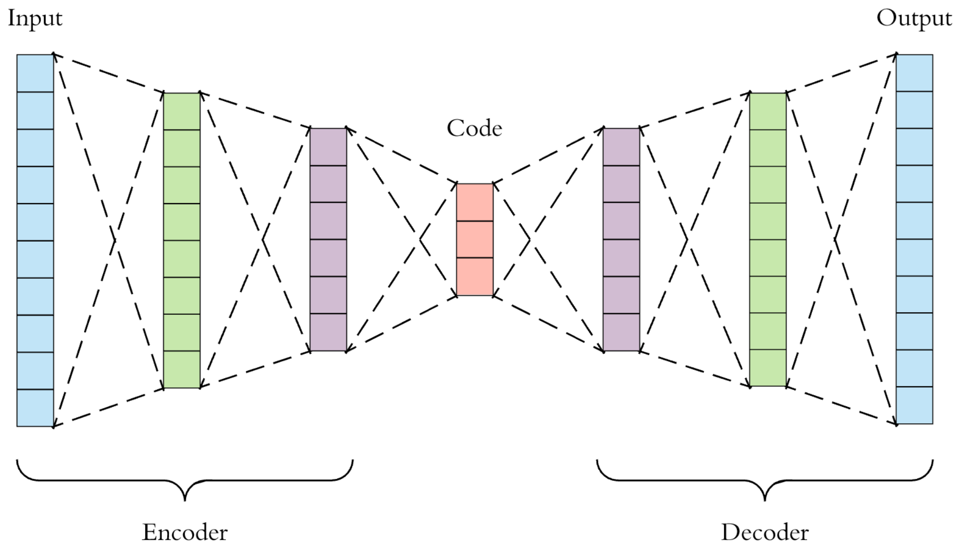


Figure 4.7: Autoencoder.

more complex relationships in sequential data. Normally another fully connected layer (non-recurrent) is attached at the end of the model for classification or prediction. We refer interested users to [167, 168] for detailed discussion on LSTM.

A stacked (or deep) autoencoder (details in [158]), as shown in Figure 4.7 is a DNN model designed to learn efficient data representation (or encoding) in an unsupervised way. It learns to compress the data from the input layer into representations, and then reconstructs the original data using the representations at the output layer. This process forces the autoencoder to extract the most useful features of the data.

Anomaly Detection. The above predictive models enable anomaly detection without prior knowledge of anomaly. The intuition is that since each model is trained using normal spectrum data, it *cannot* accurately predict data that contains anomaly, leading to large model prediction errors that trigger the anomaly detection. Figure 4.8 plots the anomaly detection process. We first train the model using spectrum observations in absence of anomaly, where given past values, the model predicts the next few values in the sequential data. Next, given a present spectrum observation, we use the model (and

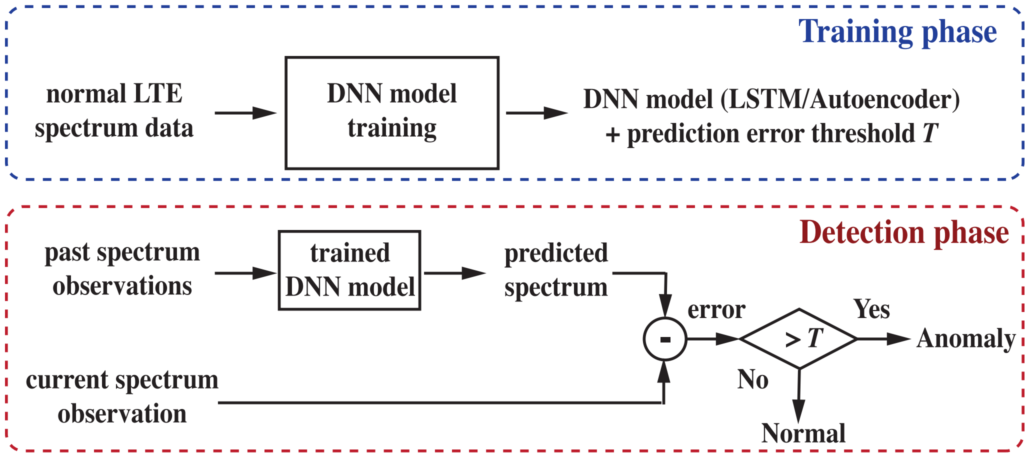


Figure 4.8: Anomaly detection using DNN models of spectrum usage.

past observations) to predict the present spectrogram, and compare it to the observed spectrogram. If the prediction error is larger than a threshold (details in §4.6), an anomaly is present.

Finding Clean Training Data. Ideally the model should be trained with measurements in absence of anomaly, which are hard to verify in practice. Fortunately, several works have confirmed that RNN, particularly LSTM can tolerate limited presence of anomalies in the training data without affecting anomaly detection performance [169, 170]. Under our target scenario, the system can choose training data from trusted observers who did not observe notable cellular service degradations at the time of data collection, thus the mass majority of the training data are in absence of anomaly.

4.4.2 Unified Models of Spectrum Usage

We now describe the unique steps we take to build and train a per-cell, unified predictive model on spectrum usage. While our description below is for LSTM, we apply the same process to build and train the deep autoencoder model.

Input to LSTM. We feed the raw spectrogram of the wireless signal into the LSTM model. Our intuition is that sufficiently powerful LSTMs operating on raw signals can extract meaningful patterns, while an alternative LSTM operating on aggregate statistics is vulnerable to poor choice of statistics, and could miss valuable dimensions of the data.

Given our LTE measurements, we configure the LSTM model to use $x = 25.6$ms of measured signal as input to predict the next $y = 6.4$ms. We chose these parameters because our spectrum analysis in §4.2 shows that the longest periodic pattern occurs at 6ms, thus a target frame of $y = 6.4$ms should be sufficiently large to include all the key patterns. We also experimented with other x values and found that 25.6ms offers the best performance under our target scenarios. We leave the optimization of x and y to future work.

Making the Model Context-agnostic. Our predictive model is context-agnostic, so that it can be deployed on all observers in the current cell, regardless of their physical location and mobility status. We take two steps to achieve that.

First, we apply *linear transformation* to expose the intrinsic spectrum usage patterns. As mentioned earlier, the input signal data displays a large variance across observing locations, which is an inherent property of radio propagation. Such high variance can cause LSTM (and autoencoder) to miss detailed temporal patterns and correlations among sub-frequencies, but focus solely on absolute power values. To expose these intrinsic spectrum patterns, we apply linear transformation, *i.e.*, mean-centering and scaling, to the input FFT amplitudes, and filter out input sequences that only contain noise (no signal at all). As a result, each input sequence to the LTSM model now has a zero mean and a variance of 1. This transformation is similar to the idea of *contrast stretching*, a common pre-processing technique in computer vision that exposes patterns by transforming pixel intensities to increase contrast [171].

Second, we use as training data a mixture of spectrum measurements collected by both static and mobile observers within the cell. Compared to context-specific models, this certainly minimizes the model training time and data requirements. One concern is that mixing training data from many sources can potentially increase the ambiguity between normal data and anomalies. For example, the normal spectrum usage seen at observer A could be similar to the anomalous spectrum usage seen by another observer B. When the training data includes those measurements from A, the trained model could misclassify B's observation of an anomaly as normal.

We took a detailed look at our measurement data (with anomalies), but did not identify any of such events. While our anomaly instances are limited in scale, this result suggests that the probability of such events is low in practice. An advanced attacker could form its misuse signals to imitate normal spectrum usage, but this is challenging since the observers will observe the aggregated signals from the attacker and the legitimate LTE transmitters.

Model Training. To train these models, we divide our per-cell LTE measurement data into two portions: one used for training, and one for testing. Both datasets contain measurements collected by static, walking and driving observers. The observers in the testing dataset do not appear in the training dataset. Overall, for each cell, the ratio of the training data volume and the test data volume is 2.5:1. Each model is trained to minimize the Root Mean Square Error (RMSE) between the predicted signal and the ground-truth signal (after transformation) in the training dataset.

4.4.3 Evaluation: Model Prediction Error

We evaluate how well the DNN models predict the spectrum usage of the immediate future, so that they can quickly detect anomalies that disturb the spectrum usage pattern. As an illustrative example, Figure 4.9 plots the actual spectrogram (on a randomly chosen

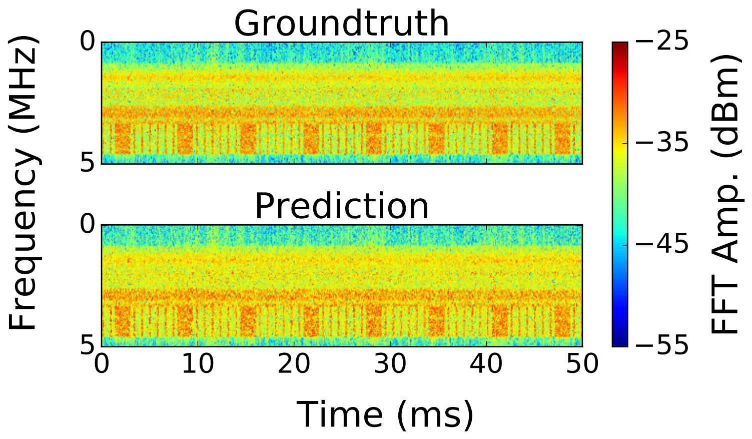


Figure 4.9: Spectrograms of actual and LSTM predicted LTE signal.

50 ms segment) and the output of the LSTM prediction model (after reversing the linear transformation). To reconstruct the 50ms segment from the prediction result, we cascade 8 segments of 6.4ms prediction results, each predicted from previously observed 25.6ms measurements. We see that the LSTM model is able to recover the key patterns in spectrum usage across sub-frequencies and time.

Evaluation Metric: Spectrogram Prediction Error (dB). We evaluate each model by the difference between the true signal spectrogram and the model prediction. Specifically, we calculate the RMSE between the true FFT amplitude (dBm) across sub-frequencies of the LTE band and the LSTM model prediction values after being inverse-transformed back to FFT amplitude (dBm). In a nutshell, the RMSE value approximates the amplitude spectrogram error in a single dB value. We refer to this metric as the prediction error (dB). Because our test data has many observers, we will present the mean and standard deviation of the prediction error across all the observations in the test dataset.

The prediction error directly links to the accuracy of anomaly detection. The smaller the prediction error is in absence of anomaly, the better the predictive model is and the higher accuracy the model has during anomaly detection. We evaluate the anomaly detection performance later in §4.6, which yields consistent results.

Unified vs. Customized Models. We evaluate our unified models by comparing them to models customized to individual observer's context. The results of LSTM and autoencoder are similar to each other: the prediction error of autoencoder is 3-8% higher than that of LSTM. We only show the LSTM results for brevity.

Table 4.1(a) shows the mean and standard deviation of the prediction error (dB) of our unified model and those of the models customized to three individual locations. The location-specific models, when running on a different location, produce large prediction errors (5.21 dB rather than 2.5dB). But the unified model is always as good as or even better than all the location-specific models.

We repeat the experiment in the time domain. Table 4.1(b) confirms that training data over day and night can also be mixed together when building the unified model. We also use data collected in June 2018 to further test our model (trained using measurements from January and March 2018), and the unified model consistently provides better prediction than those designed for specific time periods of the day. The difference between the models is less visible compared to that in Table 4.1(a), indicating that physical location has a much heavier impact on spectrum monitoring than time.

We also experiment with the mobility context. We group the measurements by their mobility context: static (mixed locations), walking, and driving ($\leq$ 25 mph). In addition to the unified model, we also trained mobility-specific models for each of the three contexts. Table 4.1(c) shows that the unified model and the mobility-specific models perform similarly. For both, the average prediction error is bounded by 2.62 dB with a very low variance (0.26).

Overall, the unified model achieves the best prediction performance, 2.58 dB (0.19), when tested at a diverse set of observers. This can be attributed to two factors. First,

Test \ Train	Loc 1	Loc 2	Loc 3	Unified
Loc 1	**2.71 (0.38)**	5.21 (12.74)	2.97 (0.56)	2.67 (0.32)
Loc 2	3.41 (0.64)	**2.46 (0.76)**	3.67 (1.09)	2.47 (0.24)
Loc 3	2.68 (0.46)	4.56 (7.03)	**2.63 (0.26)**	2.61 (0.29)
Mixed	3.70 (1.79)	4.86 (13.84)	4.07 (1.88)	**2.59 (0.23)**

(a) Across Location

Test \ Train	Day	Night	Unified
Day time	**2.59 (0.23)**	2.74 (0.22)	2.53 (0.18)
Night	2.63 (0.23)	**2.71 (0.27)**	2.61 (0.22)
Mixed	2.59 (0.18)	2.74 (0.18)	**2.55 (0.22)**

(b) Across Time Periods

Test \ Train	Static	Walking	Driving	Unified
Static	**2.52 (0.21)**	2.47 (0.23)	2.57 (0.33)	2.43 (0.23)
Walking	2.47 (0.29)	**2.62 (0.23)**	2.67 (0.27)	2.56 (0.18)
Driving	2.64 (0.29)	2.58 (0.26)	**2.59 (0.26)**	2.57 (0.23)
Mixed	2.66 (0.30)	2.54 (0.27)	2.61 (0.23)	**2.58 (0.19)**

(c) Across Mobility Context

Table 4.1: Prediction error (dB) of LSTM models under different training configuration. Numbers in parenthesis show the standard deviation of prediction error (dB). Here we show the result from the 880MHz band while the other bands lead to similar conclusions.

the model's timing configuration (using 25.6 ms data to predict next 6.4 ms data) allows LSTM to capture critical spectrum usage patterns, and yet remains small enough to make the model robust against context changes. Second, the linear transformation allows LSTM to focus on intrinsic patterns of signal spectrogram, which remains consistent across different mobility context, time periods, and locations.

It should be noted that a related challenge is whether and how such unified model per LTE cell can be used near cell boundaries, where an observer can potentially pick up signals from multiple basestations. When these basestations operate on the same frequency band, the observer could see signal patterns that are different from those at in-cell locations. This must be treated with care to minimize false alarms. As future work, we plan to address this issue using dedicated measurements at cell boundaries.

Model Complexity. We implement our LSTM and autoencoder models on a NVIDIA Titan X GPU, where it takes < 10ms for prediction on each data segment. As future work, we plan to implement our design on commodity mobile platforms such as smartphones and NVIDIA Jetson platforms. Existing works have successfully deployed efficient LSTM models on mobile devices [172, 173]. The LSTM model in [173] has 5 layers, each with 500 LSTM units, and runs efficiently on Nexus 5 Android smartphones. In comparison, our LSTM has fewer parameters (2 LSTM layers, 64 units each) and should also run efficiently on common mobile devices.

4.4.4 Models for DL and UL Bands

We take a closer look at the unified models built for each of the four LTE bands. Recall that our analysis in §4.2.2 shows that LTE bands display different spectrum usage patterns. The UL band (830 MHz) is particularly different from the DL bands

Interestingly, the final model structure also differs between the DL and UL bands. For LSTM, the three DL bands share the same structure: 2 LSTM layers of 64 units plus 1 dense layer, while the UL band requires an extra LSTM layer. The same applies to autoencoder: the DL models have 4 dense layers while the UL model has 6 dense layers. This is somewhat intuitive since LTE DL signals originate from a single strong transmitter (basestation), while UL signals are aggregates of many weak transmitters. The UL spectrum patterns are more complex, requiring more neurons to learn.

Table 4.2 lists the prediction errors of the four bands using LSTM. The three DL bands perform similarly, while the UL band experiences larger prediction errors. We also verified the same model using spectrum measurement data collected a few months later

(June 2018), and the results are consistent.

	880 MHz	750 MHz	730 MHz	830 MHz (UL)
Testing: Early 2018	2.58 (0.19)	2.70 (0.25)	2.54 (0.26)	3.14 (0.78)
Testing: June 2018	2.61 (0.21)	2.72 (0.24)	2.52 (0.25)	3.11 (0.73)

Table 4.2: Model prediction error (dB) of our unified LSTM model.

4.5 Beyond the Per-Cell Model

We now consider the problem of building spectrum models for many LTE cells and multiple bands. One can simply train a model for each LTE cell and band, but the training overhead and data collection requirements are practically prohibitive. Like other DNN models, LSTM and deep autoencoder require large amount of diverse training data and can take hours and even days to finish training. Currently, our models take 2 hours to finish with 1 day worth of training data and almost 24 hours with 8 days of training data. Practical deployment will likely need much larger and more diverse training data, and more frequent training to adapt the models. Thus it is critical to reduce the model training overhead across all the LTE cells. In the following we discuss and compare potential solutions to address the issue of training overhead.

4.5.1 Can Models be Reused?

Does reusing the model across cells and bands really work?

Test I: Reusing Models across LTE Cells. We apply the LSTM model trained for one cell to another (same network carrier, same frequency band, same technology, just a different basestation), and observe sizable performance degradation in both model prediction and anomaly detection. For the 880 MHz DL band, the average prediction error raises to 3.36 dB from the baseline of 2.58 dB and the standard deviation jumps from 0.19 to 0.56. This is likely because the two basestations are configured differently so their spectrum usage patterns differ.

Test II: Reusing Models across LTE Bands. We apply the LSTM model trained on the 880 MHz DL band to the 730 MHz DL band. The average prediction error and the standard deviation grow to 3.54 dB (0.71) compared to 2.70 dB (0.25). This is because the two bands show visible differences in spectrum usage patterns, which are captured by LSTM to produce a precise model for each. Autoencoder shows the same trend.

Together, these results show that models trained for a specific LTE cell and a specific LTE band are in general *not* reusable across cells and bands. This does not contradict with our conclusion in §4.4 where the per-cell model can be reused within the same cell. In a given cell, the spectrum usage pattern is fairly consistent.

4.5.2 Fast Training via Transfer Learning

To speed up model training at many cells, we consider an alternative solution, *transfer learning* [174], which adapts a pre-trained DNN model to a new scenario using limited training data. It leverages the underlying similarity between tasks associated with two models. By transferring model architecture and weights from a pre-trained model (*teacher*) to the new model (*student*), one can bootstrap and fine-tune the student model with limited training data.

Transfer learning is suitable for our problem because LTE cells share similar spectrum usage characteristics (§4.2), especially for DL bands since only LTE base stations are transmitting. Next we show that transfer learning can be used to quickly adapt a pre-trained LSTM model to a new LTE cell and even to a new LTE band, reducing training data volume by a factor of 288.

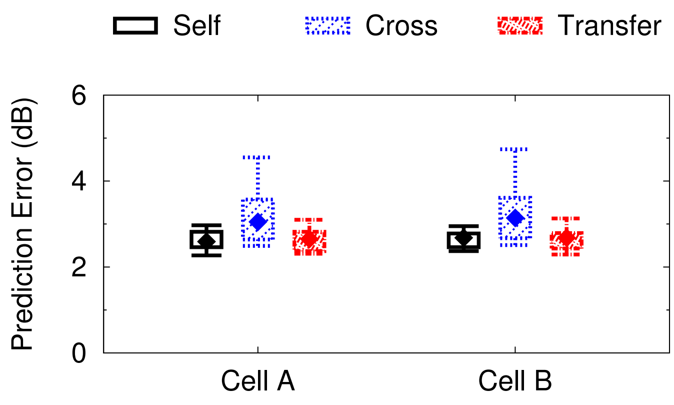


Figure 4.10: Prediction error of LSTM models transferred across different LTE cells in the 880 MHz DL band.

We note that a similar concept of "knowledge transfer" has been applied to wireless networking design, using knowledge collected by one basestation to help configure another basestation (*e.g.*, spectrum handoff [175], operating modes for energy saving [176] and content caching strategies [177]). Our solution is motivated by these existing works, but our contribution lies in the *novel application* of transfer learning to the problem of spectrum anomaly detection, and a detailed validation using real-world LTE measurements.

When applying transfer learning, we first build a student model by copying the teacher model, then use local spectrum measurement data to refine the model. Here the transfer process depends on k, the number of model layers "allowed" to be updated [174]. The simplest form of transfer learning updates on the last (dense) layer of the model. This is commonly used when the student model targets very similar task or domain characteristics as the teacher model. The more advanced ones allow more or all the layers to be updated. The student models can better adapt to new scenarios but require more local data to reach convergence.

Transfer across LTE Cells. To test the effectiveness of transfer learning, we transfer

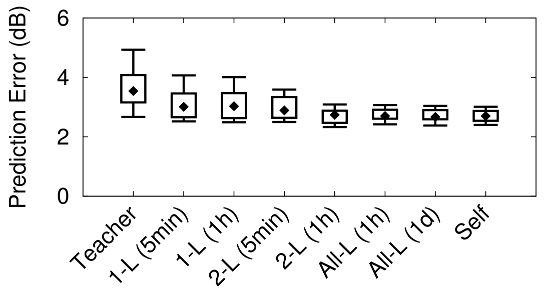


Figure 4.11: Prediction error of LSTM models with different transfer options (transfer model of DL 880 MHz to DL 730 MHz).

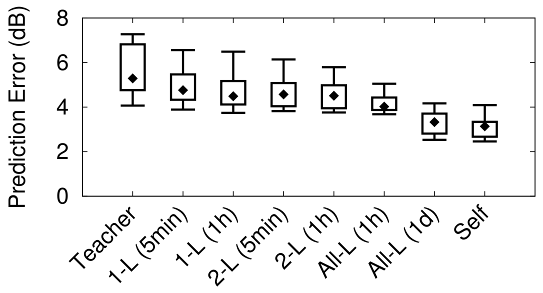


Figure 4.12: Prediction error of LSTM models with different transfer options (transfer model of DL 880 MHz to UL 830 MHz).

the LSTM model trained on cell A (using 1 day of training data) to cell B (same carrier, band, technology), and fine-tune B's model with 5 minutes of spectrum data collected in cell B. Here we chose 5-min because the tuning process converges. We consider the simplest transfer option of freezing all weights of the two LSTM layers and only updating weights in the last dense layer. For comparison, we train another LSTM model for B from scratch using the same amount of training data as of A (1 day).

Figure 4.10 shows the prediction error of the model trained from scratch (Self), the model of the other cell (Cross), and the transferred model (Transfer). The transferred model's error mean (2.65 dB) is extremely close to that of the model trained with the full data (2.59 dB). Yet this model only requires fine tuning the last dense layer using 5-min local spectrum data, comparing with 1-day worth of data for Self (a factor of 288 reduction). The same conclusion holds when we test the autoencoder model.

Transfer across LTE Bands. Since different LTE bands display different spectrum patterns, we expect more efforts to complete the transfer learning. Since our LSTM model has three layers, we experimented with three transfer approaches: 1L, 2L, All-L, respectively, to reflect the number of model layers it needs to fine tune. We also include the results of copying the teacher model (Teacher) and training from scratch (Self).

Figure 4.11 shows the quantile distribution of the LSTM model prediction error (in absence of anomaly) by transferring the model of the DL 880 MHz band to the DL 730 MHz band, with different transfer options. Since the underlying temporal patterns differ between these two bands, only fine-tuning the last dense layer is insufficient (the average prediction error is 3.13 dB compared to 2.70 dB of Self) even after adding more training data. In the end, fine tuning 2 layers with 1 hour of data achieves comparable performance of Self. Interestingly, fine-tuning all 3 layers with 1 day worth of training data slightly outperforms training from scratch (Self). Again the same trend applies to the autoencoder model.

We also seek to transfer the DL model (*e.g.*, 880 MHz) to the UL band (830 MHz). As mentioned in §4.4.4, when trained from scratch, the 830 MHz band requires more dense layers for both LSTM and autoencoder than the three DL bands. Thus direct transfer between the two types of bands might not be as effective as the above case. Figure 4.12 confirms that for LSTM, even after fine-tuning all the layers (of the transferred 880 MHz model), the prediction error is still not on par with that of Self (which needs an extra LSTM layer). Therefore, while transfer learning can potentially be applied to quickly customize LSTM (and autoencoder) models across bands, choosing the right teacher model can be a critical requirement. We leave this topic to future work.

Complexity. We implemented the transfer learning process on the NVIDIA Titan X GPU server. Across cells, the re-training took less than 5 minutes, and uses 5-minute of local data. Across bands, the re-training took less than 2 hours, using 1-day worth of local measurements. For both, training from scratch would take 24 hours, and use 8-day worth of spectrum measurements.

4.6 Evaluation: Anomaly Detection

In this section, we use real anomaly instances to evaluate our anomaly detection system built on the DNN models. Our evaluation seeks to answer the following questions: (1) whether our unified model performs as good as the (impractical) oracle system that builds context-specific model for each location, (2) how models trained via transfer learning perform in anomaly detection compared to those trained from scratch.

Choosing Anomaly Threshold. Our DNN models detect an anomaly if the difference between the measured data and the model predicted data exceeds a threshold. The threshold also determines the false alarm rate. To determine this threshold, we partition the model training data to two subsets: the training set and the validation set. After a model is trained, we use the validation data to calculate the statistics of the model prediction error. For our dataset, these errors can be modeled using a Gaussian distribution. From this distribution we can calculate, for each false alarm rate, the corresponding threshold on the prediction error.

Next, we discuss our experiments using two types of anomalies: unauthorized transmitters and misconfigured LTE basestations. For all the experiments, the detection rate of autoencoder is similar to that of LSTM (only 2-4% worse while keeping the same false alarm rate). Thus we only show the LSTM result for brevity.

4.6.1 Detecting Unauthorized Transmissions

We generated anomalies in the form of unauthorized spectrum usage, where an "unlicensed" transmitter (USRP N210) broadcasts various types of signals. Our anomaly instances include transmissions using the entire 5MHz band and OFDM signals (like LTE), using a portion of the 5MHz band (1, 2, 3MHz) with OFDM signals, and a narrowband misuse with QPSK and BPSK signals. When the anomaly is on, we set up a static observer and a walking observer in proximity (within 50 meters) who collect LTE measurements and perform anomaly detection. The walking observer follows a predefined route for all the anomaly instances. We did not have any driving observers since they quickly go out of range of our low power transmitters. In total, we performed 100 experiments, each of 5 minutes long.

Ethics. We are very aware of the potential impact of our experiments on cellular users, and took extensive precautions to ensure that these experiments had no impact on cellular users or basestations. First, we chose the second floor of an older campus building with heavy concrete walls and floors as our setting. We first measured signal propagation properties in the building by setting the transmitter frequency to 900MHz (closest unlicensed band), and using a spectrum analyzer to measure signal strength at numerous locations inside and outside of the building. We confirmed that the thick concrete walls and floors completely blocked signals beyond the immediate open hallway and adjoining offices and no signals were observed outside the building or on floors above or below. Next, we scheduled experiments late at night and on weekends, when the campus building is generally unoccupied. Between experiments, one student walked the entire length of the hallway and checked to make sure no one else is on the floor. We did not encounter any other occupants of the building during our experiments. We also periodically used spectrum analyzers to (re)confirm that our transmissions are strictly constrained to the second floor.

Results: Anomaly Detection Accuracy. Figure 4.13 compares the quantile distribution (5%, 25%, 50%, 75%, 95%) of the model prediction error (dB) per spectrum observer, *i.e.* the RMSE between the measured and predicted spectrograms, across all the measurement instances, with and without anomalies. The presence of anomalies largely

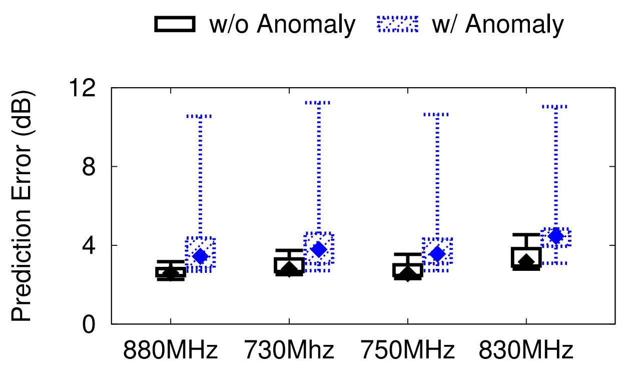


Figure 4.13: Quantiles of model prediction errors w/ and w/o anomalies.

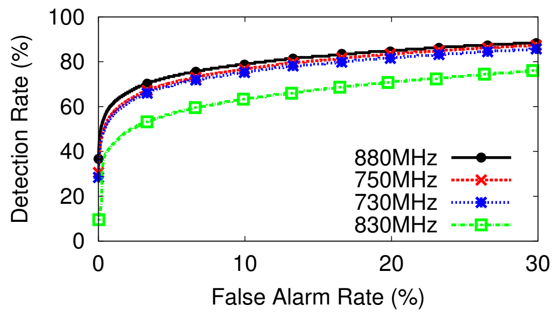


Figure 4.14: Detection rate vs. false alarm rate of unauthorized transmitters of different bands.

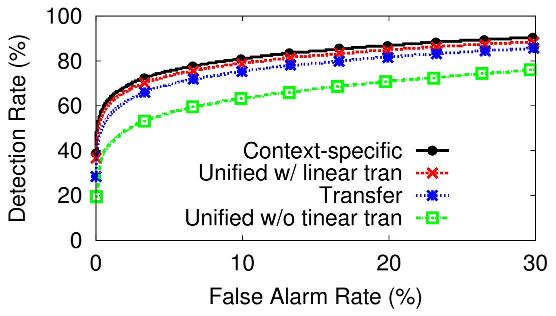


Figure 4.15: Our unified and transferred models are on par with the oracle design that uses location-specific models.

increases the prediction error. The two distributions are reasonably separated for the three DL bands, but overlap slightly for the UL band (830MHz). Next, Figure 4.14 plots the RoC result (anomaly detection rate vs. false alarm rate) for the four LTE bands. Here we average the detection result across all the measurement instances collected by the static and mobile observers, producing the average detection rate per spectrum observer. We see that for the three DL bands, the anomaly detection results are on par with each other, while the UL band is less effective. Yet these results are still significant better than non-DNN solutions (see Figure 4.4).

In our experiments, misdetection occurs when the anomaly's signal power is low and the observer is further away from the misuse. In practical deployment, one can improve the anomaly detection rate by deploying more observers for density and coverage, further pushing the need for a context-agnostic model.

Results: Unified vs. Customized Models. We compare our unified model (with linear transformation) and the basic version (without linear transformation), to an context-specific system that builds context-specific models for each location. We compute the anomaly detection result for the context-specific system by training a model for each static observer using its own past observations, and testing the model using the anomaly instances in range of the static observer. Note that our unified model is tested on all the anomaly instances and on both the static and mobile observers. As shown in Figure 4.15, our unified model (with linear transformation) performs as well as the context-specific model.

Results: Transferred vs. Self-trained Models. We compare the anomaly detection performance of models transferred from other cells with those of models trained from scratch. Figure 4.15 shows that the transferred model achieves almost identical performance while greatly reducing the training overhead.

4.6.2 Detecting TX Misconfigurations

We also study anomalies of basestation misconfiguration, implemented by modifying our LTE measurement traces. This will not produce any impact on cellular services. We consider two types of misconfiguration: (1) the misconfigured basestation stops transmitting signals at some or all sub-frequencies; (2) the misconfigured basestation suddenly changes its transmit power level. We produce both instances by modifying our LTE downlink measurement traces, replacing them with replays of measured noise signals or increasing/decreasing the amplitude of the received signals.

We note that these anomalies could also be detected by other methods, since (strong) static observers will likely detect changes in the spectrogram. Instead, we use these to show that our unified model can detect *general* types of anomalies beyond unauthorized transmissions. That is, the *same* model can detect both unauthorized transmissions and misconfiguration of basestations.

	ΔP=3 dB	ΔP=5 dB	33% F down	66% F down	100% F down
880 MHz	58%	78%	52%	87%	100%
750 MHz	60%	81%	57%	90%	100%
730 MHz	53%	78%	54%	88%	100%

Table 4.3: Anomaly detection rate at 1% false alarm rate.

Detection Results. Table 4.3 lists the average detection rate per spectrum observer under 1% false alarm rate for different categories of misconfiguration. Here "x% F down" means transmissions on x% of frequency is replaced as noise. "ΔP =x dB" means transmit power is modified by x dB. Even at a very low 1% false alarm rate, the same unified model (as in §4.6.1) can effectively detect anomalies caused by misconfiguration, and the detection rate correlates with the severity of the anomaly. While linear transformation used to build our model "suppresses" the impact of transmit power level, our model can still detect sudden basestation power changes because it creates notable changes in the spectrum usage pattern.

4.7 Recognizing Anomaly Types

In addition to detecting anomalies on the fly, our LSTM based approach can potentially recognize each individual type of anomalies, by examining the temporal pattern of the prediction error.

Spectrum Misuse. Figure 4.16 shows a sample event where the misuse starts at 4s and the LSTM prediction error elevates immediately (from 2dB to 8-30dB), indicating

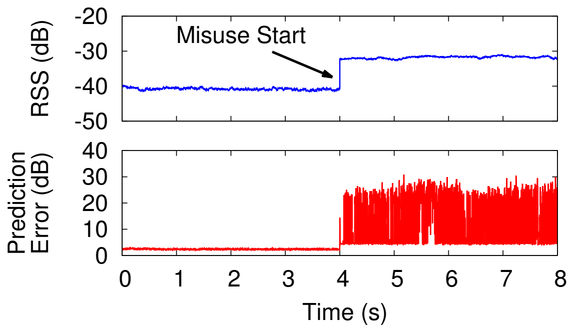


Figure 4.16: Unauthorized TX.

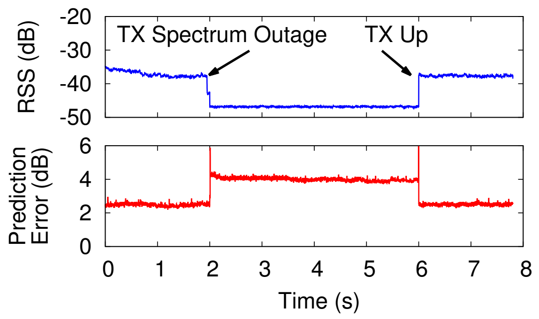


Figure 4.17: Misconfigured LTE transmit frequency.

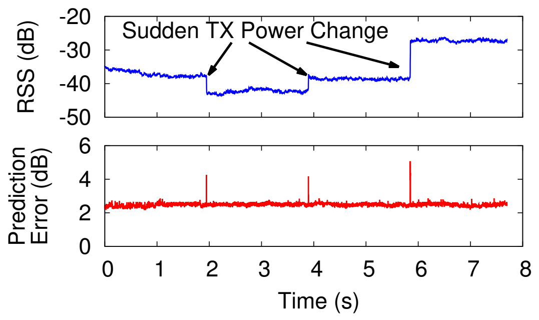


Figure 4.18: Misconfigured LTE transmit power

an anomaly event. As long as the misuse is present, the prediction error remains elevated and displays a large variance over time. Therefore, an observer can detect this anomaly by taking a spectrum measurement at any given time.

Misconfigured LTE Transmission Frequency. Figure 4.17 shows a randomly chosen example, where the basestation enters an outage at time 2s and recovers at time 6s. These two sudden changes trigger two sharp spikes (immediately) in the LSTM prediction error, which remains elevated during the outage. Compared to the misuse scenario that introduces an extra transmitter, the mean and variance of prediction error during the frequency outage are relatively smaller. On the other hand, the unique feature of the prediction error (a spike followed by elevated but stable values) can be used to distinguish this anomaly from the misuse ones.

Misconfigured LTE Transmit Power. Figure 4.18 shows an example that includes three events, each with a different amount of power change. Here the LSTM prediction error displays a spike at the time of sudden power change, but falls back to normal values (< 3 dB) immediately. This means that to detect such anomaly, the observer will need to monitor the spectrum band continuously. We note that such anomaly is inherently harder to detect without continuous monitoring, as transmissions post power change still display similar spectrum usage patterns. By simply looking at each instantaneous observation, such change in power can also be caused by moving to a new location or human body blockage.

The above benchmarks indicate that each anomaly type displays unique patterns (Figure 4.16–4.18). Furthermore, the amplitude of the prediction error also shows strong correlation with the strength of the anomaly, which can be used to facilitate fault diagnosis and localization.

The same task is challenging by just observing the RSS values. For example, the RSS patterns of the unauthorized transmission in Figure 4.16 and the sudden rising of LTE transmit power in Figure 4.18 are indistinguishable. But the corresponding LSTM prediction errors are largely different.

4.8 Related Work

Anomaly Detection and Diagnosis in Wireless Networks. Existing works can be divided into three categories, depending on who runs anomaly detection and diagnosis. The first category involves system administrators. Many [178, 179, 180, 181, 131] use system logs or Key Performance Indicators (KPIs) to detect network outages and performance degradations [182]. The second category uses diagnosis by network clients. WiFiProfiler [183] studied 802.11 fault detection using information of client's wireless configuration and measurement data from the transport layer and above. The third category uses third-party devices to monitor and detect physical layer anomalies such as spectrum misuse, using metrics like signal strength variations [184, 185].

Our work falls into the third category. Our key contributions are the novel application of DNN (LSTM and deep autoencoder) and transfer learning to the problem of spectrum anomaly detection, the design of a context-free DNN model, and the empirical study using measurements by both static and mobile sniffers.

Misuse Detection for Opportunistic Spectrum Access. Existing works have studied the issue of spectrum usage violation where a secondary user tries to transmit when a nearby primary user is inactive. They consider individual features of signal transmissions, including average received signal strength distribution over space [160, 161, 162, 186, 187, 188], signal strength variation [163, 164], physical channel properties [166, 189, 190], signal amplitude difference between direct and reflected paths [191] as well as airtime utilization [159]. Most of these designs were based on abstract propagation models, which do not capture real world settings.

Our work considers the issue of spectrum anomalies due to unauthorized transmitters and misconfiguration of LTE basestations. Our work differs from existing works by taking an empirical, data-driven approach. Instead of relying on a fixed set of features, we build DNNs to automatically extract the features required for accurate anomaly detection, and develop context-free DNN models.

Machine Learning for Signal Classification and Anomaly Detection. Early work on anomaly detection focused on statistical hypothesis test [159] and threshold-based methods [184, 161]. Recently, ML models have been applied to the problem of signal classification (*e.g.* [192, 193]) and spectrum misuse detection [157, 162, 158, 194]. [157] used a small scale study to show that LSTM outperforms Kalman sequence predictor. [158] developed an Autoencoder model for spectrum anomaly detection, based on a limited dataset (1000 samples) on the FM band. [162] applied one-class SVM to detect spectrum anomaly (via simulations) while [194] used supervised learning to train Hidden Markov Models.

These existing works focused on anomaly detection by a single static observer without considering the impact of user mobility and location. They used either simulation or few measurement data for validation. Our work has a much broader scope by developing robust, scalable anomaly detection capable of detecting previously unknown anomalies, for both static and mobile observers. We also collected detailed signal measurements on four LTE bands and under different user context to drive our empirical study.

4.9 Conclusion & Future Work

We show that a scalable DNN model on LTE spectrum usage can be built and deployed on a wide range of observers (static nodes, walking users, buses), enabling real-time spectrum anomaly detection. Its performance matches the "oracle" design that trains customized models for each specific user context (location and mobility). The model remains constant for any observer in a single cell, and can be quickly trained and adapted using a small amount of local spectrum measurements. To the best of our knowledge, this is the first to show the feasibility of building practical and general spectrum anomaly detection systems for large-scale LTE networks.

Moving forward, we plan to explore several directions. *First*, we plan to explore other DNN models, and validate them using spectrum anomalies in the wild. Also, the frequency-temporal pattern of the prediction error could be used to distinguish different anomaly types. This needs to be further validated using anomaly instances in the wild. *Second*, spectrum measurements at individual observers can be noisy [155], or corrupted/modified by well-equipped adversaries. We plan to refine our system to be robust against such artifacts. *Finally*, the spectrum usage may change over time, *e.g.*, upon carrier upgrade. While our current measurements did not observe such changes, how to update the model to the new configuration will be an interesting direction to explore.

Chapter 5

Conclusion

In conclusion, this thesis presents the use of data-driven network design to solve practical networking issues. We focus on addressing the challenges caused by limited, noisy and biased measurement data, and those arisen from making the machine learning model scalable. We propose three data-driven designs in the area of RF transmitter localization, network anomaly detection and spectrum anomaly detection. Our proposed systems lead to significant improvement in network services.

This chapter summarizes the contributions of our work, along with discussions on open problems on data-driven network design.

5.1 Summary of Contributions

Recently, data-driven paradigm has been applied to solve complicated tasks in various areas. In general, data-driven paradigm allows the solutions to be built directly on the data produced in the tasks. Thanks to the rapid development of network monitoring tools over the years, we are able to obtain a large amount of network measurement data that makes it feasible to apply data-driven paradigm in network management. While prior studies have demonstrated the initial feasibility of data-driven designs, applying them to practical systems designs still faces significant challenges. This dissertation proposes data-driven designs in three real-world network tasks, focusing on tackling three practical challenges:

- *Optimize input data* - network measurement data can be unreliable, we propose feature clustering to identify data values. By filtering out the data of bad quality, we can able to improve the network performance.
- *Optimize learning algorithms* - network/spectrum anomalies are hard to be captured due to the small event volume (network data bias), we design a special machine learning algorithm to deal with that.
- *Scale deep learning models* - data pattern changes across different context (*e.g.*devices, time and location), we apply advanced machine learning techniques to make the model able to scale and do fast model adaptation for better efficiency.

In Chapter 2, we present our data-driven design for RF transmitter localization. We first demonstrate the feasibility of deploying distributed spectrum monitoring system (with RF transmitter localization as the major task), which is built on commodity smartphones and embedded low-cost spectrum sensors. In general, the network providers pay non-network users to do the measurement tasks, creating a large amount of the crowdsourced measurements. We observe that the crowdsourced measurements can be unreliable due to factors of hardware, environment and human artifacts. This will significantly degrade the localization accuracy. To solve the problem, we optimize the input data of each localization task. Thus in the second part, we studied how to identify values in crowdsourced wireless signal measurements using large-scale datasets. We propose *feature clustering*, a novel application of unsupervised learning to detect hidden correlation between measurement instances, their features, and localization accuracy. This approach enables us to recognize the key features, and use the key features to identify the types of measurement data that correlate well with high or low prediction accuracy. By only using the measurements of high prediction accuracy, we are be able to improve the localization accuracy.

In Chapter 3, we show how we apply data-driven approach on system log data to achieve network trouble prediction for Network Function Virtualization (NFV). We propose a deep neural network model utilizing Long Short-Term Memory (LSTM), to model the system log as a natural language sequence. It can automatically learn log patterns from normal execution, and detect anomalies when the log patterns deviate from the model trained from log data under normal execution. The abnormal log patterns will be treated as the trigger predictions of network faulty conditions. Our proposed methodology enables to easily scale the model across different NFV hosts and infrastructure changes, without incurring extended delays for collecting training data. We evaluate our methodology using 18-month real-world deployment data provided by AT&T on virtualized provider edge routers. The results show that the detected anomalies can help the operators understand the faults, and optimize the trouble ticket generation rules enabling faster, or even proactive actions against faulty conditions.

In Chapter 4, we explore the design of a general, scalable system for detecting spectrum anomalies in wide-area LTE networks. The system is consist of 1) a scalable, distributed spectrum monitoring system that measures physical spectrum usage using both static and mobile observers distributed across the network, and 2) a general anomaly detection system that builds deep neural network (DNN) models using these measurements, and runs them at each observer as baselines to detect spectrum usage anomalies. The model remains the same for any observer in a single cell, and can be quickly trained and adapted using a small amount of local spectrum measurements using transfer learning.

To the best of our knowledge, this is the first work to show the feasibility of building practical and general spectrum anomaly detection systems for large-scale LTE networks.

5.2 Future Work & Discussion

The application of data-driven paradigm in networked and mobile networked system is still in its infancy. There are tremendous opportunities for data-driven techniques to help improve network services. In this section, I present several topics towards practical data-driven designs that are related to my thesis work.

5.2.1 Address Data Issues

In the previous sections, we discussed data issues in terms of data quality and bias. In practice, there could be other types of issues in measurement data. Here, I list some common issues.

Data Imbalance. We often observe data/event imbalance in the real-world system. For example, in Section 4 and 5, we observe an extreme data imbalance between troubled events and normal events. Further more, we observe that the even for all the network trouble events, events of different root causes are also imbalanced (*e.g.*, there are much more network trouble tickets about the routing rather than the hardware). If we want to build a classification based on the measurement data, *e.g.*, determining the type of the network fault based on the log data pattern, data imbalance will be the key obstacle.

Based on how imbalanced the data is, there are different solutions. In the most extreme cases, it may be better to think of classification under the context of anomaly detection, like what we did in the previous sections. In other cases, we could consider the following solutions to deal with the problem: *First*, we can resample the dataset, oversample the minority classes or undersample the majority classes, to make the data balanced. *Second*, we can generate some synthetic samples by randomly sampling the attributes from instances in the minority class. *Third*, we can try different classifiers (with different configurations) and run an ensemble of the classifiers. What exactly is the most effective approach, it depends on the dataset.

Limited Training Data. Deep learning models are notorious for performing well only with a great amount of training data. In the low-data regime, the parameters are underdetermined, and the learnt networks generalize poorly. However, the training measurement data may be difficult to obtain for some networked tasks. There are two reasons of why the data is limited: 1) The data itself is difficult to obtain. It happens when we look at a new or unique task. For example, we want to build a classification of the health status for Parkinson's disease patients based on the wearable sensing data [195]. The data is limited due to the number of the patients is small. 2) The label of the data is difficult to get. For example, in the task of activity recognition using IoT devices, the data is easy to get since the IoT devices are very common for each home. The IoT devices monitor human's activity every day and upload the data to the server once detecting a motion. However, the data is uploaded without knowing the type of the activity. Thus we need to manually label the data afterwards, which is both time and manpower consuming. The most simple and naive way is to wait for enough labelled data. However this will make the deep learning approach ineffective and impractical.

There are several interesting ways to solve the problem. For example, *Data Augmentation* [196] is proposed to generate more data from the existing data. Generative Adversarial Networks (GANs) [197] is one of the most popular data augmentation approaches. Few-Shot Learning [198, 199] and Meta-Learning [200] are proposed to speed up the learning ability of the model, so that the model can learn new concepts and skills fast with a few training examples. Self-Supervised Learning [201] is recently proposed as a learning technique where the training data is automatically labelled. It is still supervised learning, but the datasets do not need to be manually labelled by human, but they can *e.g.*, be labelled by finding and exploiting the correlations between different input signals (*e.g.*, input coming from different sensor modalities). All of the above techniques are widely explored in the computer vision area, but currently we don't see much work apply those in the networking area. Towards building a practical (fast-learning and effective) intelligent networked system, this is a necessary and interesting direction to explore.

5.2.2 Towards Self-Driving Networking

Self-Driving Network is one of the ultimate goals for telecom network evolution. Basically it contains three parts: 1) data sensing and collecting from the network; 2) data analysis to infer characteristics about the network; 3) decision making of whether and how to adapt the network's configuration in response to changing the network conditions. Today, the three parts are performed separately on different timescales, and often in a slow or manual fashion instead of data-driven approaches. While in the future, the networks should be able to combine the three parts and manage themselves autonomously. The key point of achieving self-driving network is to learn how to sense and monitor itself, as well as program and control itself from the data. This is challenging because the network is becoming increasingly dynamic and the down-time tolerance is extremely low. We need to address all of these challenges before eventually achieving self-driving network. Here, I point out two specific directions which I'm particularly interested in.

Network Monitoring w/ Multi-Layer Measurement. In Section 4, we have shown how to build a deep learning model to monitor/infer network status using VNF syslogs in the NFV scenario. Our result shows that the circuit/routing network issues are easier to be predicted than the hardware ones. The reason of why circuit/routing network issues are easier to be predicted could be that: the syslogs we obtained is from the upper layer, and the lower layer issues (*e.g.*hardware) need time to propagate so that the warning signals we observed haven been delayed. In order to optimize the network issue predictor, we should use the monitoring data from multiple layers. But the challenge here is how to combine the data from multiple layers. We can build machine learning models for different layers separately and then combine the results, or we can directly build one machine learning model with multiple layers. However, as the monitoring data across layers may be measured at different frequency, it will make it hard to directly combine all the data. How to deal with the inputs of the model and how to design the model structure should be carefully designed.

Adaptation/Reaction Based on Network Status. Compared with network monitoring, how to automatically react or adapt to the current network status is less studied. *What to adapt/react* and *how to automatically adapt/react* are the key questions.

The adaptation can be separated in to network adaptation and application adaptation. Network adaptation includes taking actions of re-routing based on the network congestion or network failure. Application adaptation includes taking actions of adjusting the bitrate of the transmission in response to changing network conditions. A network operator may want the network and application to adapt to specific objectives. They might want the network to adapt automatically, and they may also want to help an application provider determine whether the adaptation that the application is performing is improving the performance of the application. Application providers have models for quality of experience (QoE), and they make assessments of the effectiveness of their adaptations based on these metrics. Currently, network operators do not have visibility into application QoE. The network needs better models for measuring network metrics and inferring quality of experience, but if the application provider could provide these metrics to the network directly, the network could optimize for these metrics?perhaps automatically.

The ways to achieve automation are extremely challenging. The current solutions are: when the settings are not complex, we can just use simple learning techniques to drive protocols that automatically adapt to the network conditions; otherwise, we can even leverage Reinforcement Learning [202] to drive decision making. For example, existing work [203] has already considered to use reinforcement learning in internet congestion control. However, we may not be able to achieve complete automation in a short time. Then how to combine human intervention in the middle is also a critical problem to be considered.

5.2.3 Machine Learning Security & Privacy

Security and privacy are important factors in practical data-driven network design. Here, we discuss two topics.

Privacy-preserving Machine Learning. Data-driven services require the devices to upload data. For example, the majority of the IoT devices today collect data about the user and transfer it to the cloud to get the services. This is because most of the IoT devices do not have enough resources (*e.g.*GPU) for machine learning inference or storage for saving the machine learning model. However, this may expose personal information such as location, human face and activity. Thus, privacy-preserving machine learning is recently widely studied. Basically, there are two ways to achieve that:

- *From the data side:* before uploading data to the cloud, we do data anonymization using techniques such as differential privacy [204].
- *From the model side:* we can either compress the model [205] to make it local inference, or we can split the model [206] into local side and cloud side so that only the processed information can be uploaded to the cloud.

Both approaches have the tradeoff between privacy and model accuracy. These techniques are still in their infancy and haven't been applied for real-world tasks. As for compressing the model, recently there is a work [207] showing that we have been using neural networks far bigger than we actually need. It means that within every neural network, there exists a far smaller (1/10- even 1/100-times) network that can be trained to achieve the same performance as its oversize parent. However, this is just a discovery and the authors haven't come up with a way to get smaller models for general cases. But this will definitely become a popular and critical topic towards practical data-driven network design.

Robust Model against Adversarial Attacks. In order to achieve privacy-preserving and transmit less data, the full or partial model is downloaded at the user side for some services. In this way, the model is exposed to the user, and the user can utilize it to do adversarial attacks. For example, in Section 5, we build a lightweight spectrum anomaly detection model that can be installed on commodity devices so that we can leverage crowdsourcing to do large-scale spectrum anomaly detection. Once obtaining the model, an attacker can then fool the model through malicious inputs. For example, the attacker can transmit signals of some certain patterns that the model won't be able to detect it as an anomaly. This is called adversarial machine learning [208], and it is a common issue in machine learning models. Currently there is no good explanation of why machine learning is vulnerable to adversarial attacks, and there's no available solution to deal with the adversarial attacks in general. However, currently what we can do is to make the model as robust as possible so that we can raise the attack cost (*e.g.*, the attacker needs a high-end device to transmit certain complicated signals).

Appendix A

Detailed Description of Features in Chapter 2.2

- *Dispersion*: shows the spread or spatial variability of measurements and calculates their distances to center. When weighted by RSS, the dispersion mean is defined as $\bar{d} = \frac{\sum_i d(i, center) * RSS_i}{\sum_i RSS_i}$, *center* is the estimated basestation location using Weighted Centroid. The std is $\sqrt{\frac{\sum_i (d(i, center - \bar{d}))^2 * RSS_i}{\sum_i RSS_i}}$.
- *Angular coverage*: measures how measurements distribute around the estimated center from the angular point of view.
- *Standard deviational ellipse*: measures the dispersion in two dimensions. The major axis is defined as direction of maximum spread of the distribution. The minor axis is perpendicular to major axis and defines the minimum spread.
- *Estimated path loss exponent*: by fitting the log-normal propagation model, the estimated path loss exponent shows the relationship between RSS and distance.
- *Spatial autocorrelation*: measures the correlation among one point and its relatively close points. Positive spatial autocorrelation occurs when similar values occur near one

another. Negative spatial autocorrelation occurs when dissimilar values occur near one another.

Bibliography

[1] "Network functions virtualisation (NFV); architectural framework." ETSI GS NFV 002, October, 2013.

[2] V. Cisco, *Cisco visual networking index: Forecast and trends, 2017–2022*, *White Paper* (2018).

[3] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, *Software-defined networking: A comprehensive survey*, *Proceedings of the IEEE* **103** (2015), no. 1 14–76.

[4] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, *Network function virtualization: Challenges and opportunities for innovations*, *IEEE Communications Magazine* (2015).

[5] S. Lee, H. Kim, D. Barman, S. Lee, C.-k. Kim, T. Kwon, and Y. Choi, *Netramark: a network traffic classification benchmark*, .

[6] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, *Network intrusion detection*, *IEEE network* **8** (1994), no. 3 26–41.

[7] H. Yu, K. Zeng, and P. Mohapatra, *Measurement-based short-term performance prediction in wireless mesh networks*, in *Proc. of ICCCN*, 2011.

[8] K. Winstein and H. Balakrishnan, *Tcp ex machina: Computer-generated congestion control*, .

[9] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, *Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction*, in *SIGCOMM*, 2016.

[10] J. Jiang, R. Das, G. Ananthanarayanan, P. A. Chou, V. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, *et. al.*, *Via: Improving internet telephony call quality using predictive relay selection*, in *Proc. of SIGCOMM*, 2016.

[11] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, *Big data-driven optimization for mobile networks toward 5g*, *IEEE network* **30** (2016), no. 1 44–51.

[12] F. Paisana *et. al.*, *Context-aware cognitive radio using deep learning*, in *Proc. of DySPAN*, 2017.

[13] `https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/resource-center/white-paper/wp-cio-global-pov.pdf`.

[14] M. Altamaimi, M. B. Weiss, and M. McHenry, *Enforcement and spectrum sharing: Case studies of federal-commercial sharing*, in *TPRC*, 2013.

[15] FCC, *Report and order and second further notice of proposed rulemaking*, *FCC-15-47* (2015).

[16] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, *On the resiliency of virtual network functions*, *IEEE Communications Magazine* **55** (2017), no. 7 152–157.

[17] J. Nam, J. Seo, and S. Shin, *Probius: Automated approach for vnf and service chain analysis in software-defined NFV*, in *SOSR*, 2018.

[18] O. Onireti *et. al.*, *A cell outage management framework for dense heterogeneous networks*, *IEEE Trans. on Vehicular Technology* **65** (April, 2016).

[19] "Universal Software Radio Peripheral." `http://www.ettus.com`.

[20] "Wireless open-access research platform." `http://warp.rice.edu`.

[21] Y. Hwang, "Cellular IoT Explained - NB-IoT vs. LTE-M vs. 5G and More." Leverege, Dec., 2016. `https://www.leverege.com/blogpost/cellular-iot-explained-nb-iot-vs-lte-m`.

[22] M. Antonakakis *et. al.*, *Understanding the mirai botnet*, in *Proc. of USENIX Security*, 2017.

[23] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, *DDoS in the IoT: Mirai and Other Botnets*, *IEEE Computer* **50** (2017), no. 7.

[24] `http://whitespaces.spectrumbridge.com/whitespaces/home.aspx`.

[25] "Google Spectrum Database." `https://www.google.com/get/spectrumdatabase/`.

[26] A. P. Iyer, K. Chintalapudi, V. Navda, R. Ramjee, V. N. Padmanabhan, and C. R. Murthy, *SpecNet: Spectrum sensing sans frontières*, in *NSDI*, 2011.

[27] FCC, *Second report and order and memorandum opinion and order, FCC-08-260* (2008).

[28] L. Littman and B. Revare, "New Times, New Methods: Upgrading Spectrum Enforcement." Silicon Flatirons Roundtable Series on Entrepreneurship, Innovation, and Public Policy, Feb., 2014.

[29] T. Zhang, N. Leng, and S. Banerjee, *A vehicle-based measurement framework for enhancing whitespace spectrum databases*, in *MobiCom*, 2014.

[30] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng, *Towards commoditized real-time spectrum monitoring*, in *HotWireless*, 2014.

[31] J. A. Wepman, B. L. Bedford, H. Ottke, and M. G. Cotton, *RF sensors for spectrum monitoring applications: Fundamentals and RF performance test plan, NTIA Report 15-519* (2015).

[32] O. Fatemieh, R. Chandra, and C. Gunter, *Secure collaborative sensing for crowdsourcing spectrum data in white space networks*, in *Proc. of DySPAN*, 2010.

[33] T. Zhang and S. Banerjee, *Inaccurate spectrum databases?: Public transit to its rescue!*, in *HotNets*, 2013.

[34] L. Shi, P. Bahl, and D. Katabi, *Beyond sensing: Multi-GHz realtime spectrum analytics*, in *NSDI*, 2015.

[35] D. Pfammatter, D. Giustiniano, and V. Lenders, *A software-defined sensor architecture for large-scale wideband spectrum monitoring*, in *IPSN*, 2015.

[36] T. Zhang, A. Patro, N. Leng, and S. Banerjee, *A wireless spectrum analyzer in your pocket*, in *HotMobile*, 2015.

[37] `http://sdr.osmocom.org/trac/wiki/rtl-sdr`.

[38] L. Yang, Z. Zhang, B. Y. Zhao, C. Kruegel, and H. Zheng, *Enforcing dynamic spectrum access with spectrum permits*, in *MobiHoc*, 2012.

[39] P. Sutton, K. Nolan, and L. Doyle, *Cyclostationary signatures in practical cognitive radio applications*, *IEEE JSAC* **26** (2008), no. 1 13–24.

[40] J. M. Dyaberi, B. Parsons, V. S. Pai, K. Kannan, Y.-F. R. Chen, R. Jana, D. Stern, and A. Varshavsky, *Managing cellular congestion using incentives*, *IEEE Communications Magazine* **50** (2012), no. 11.

[41] G. Hsieh and R. Kocielnik, *You get who you pay for: The impact of incentives on participation bias*, in *CSCW*, 2016.

[42] `https://www.tablix.org/~avian/blog/archives/2015/03/noise_figure_measurements_of_rtl_sdr_dongles/`.

[43] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, *Accuracy characterization for metropolitan-scale wi-fi localization*, in *Proc. of MobiSys*, 2005.

[44] S. Liu, Y. Chen, W. Trappe, and L. J. Greenstein, *Non-interactive localization of cognitive radios based on dynamic signal strength mapping*, in *WONS*, 2009.

[45] D. Han and S. a. o. Seshan, *Access point localization using local signal strength gradient*, in *Proc. of PAM*, 2009.

[46] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan, *Ecolocation: a sequence based technique for rf localization in wireless sensor networks*, in *Proc. of IPSN*, 2005.

[47] A. Savvides, C.-C. Han, and M. B. Strivastava, *Dynamic fine-grained localization in ad-hoc networks of sensors*, in *MobiCom*, 2001.

[48] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh, *White space networking with Wi-Fi like connectivity*, in *Proc. of SIGCOMM*, 2009.

[49] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat, *Learning to share: narrowband-friendly wideband networks*, in *Proc. of SIGCOMM*, 2008.

[50] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi, *GHz-wide sensing and decoding using the sparse fourier transform*, in *INFOCOM*, 2014.

[51] S. Yoon, E. Li, S. C. Liew, R. R. Choudhury, I. Rhee, and K. Tan, *QuickSense: Fast and energy-efficient channel sensing for dynamic spectrum access networks*, in *INFOCOM*, 2013.

[52] J. Laska, W. Bradley, T. Rondeau, K. Nolan, and B. Vigoda, *Compressive sensing for dynamic spectrum access networks: Techniques and tradeoffs*, in *DySPAN*, 2011.

[53] A. Chakraborty and S. R. Das, *Measurement-augmented spectrum databases for white space spectrum*, in *CoNEXT*, 2014.

[54] N. Brouwers and K. Langendoen, *Will dynamic spectrum access drain my battery?*, *Embedded Software Report Series, ES-2014-01* (2014).

[55] P. Kaligineedi, M. Khabbazian, and V. Bhargava, *Malicious user detection in a cognitive radio cooperative sensing system*, *IEEE TWC* **9** (2010), no. 8 2488–2497.

[56] L. Song, Y. Chen, W. Trappe, and L. Greenstein, *ALDO: An anomaly detection framework for dynamic spectrum access networks*, in *INFOCOM*, 2009.

[57] Z. Chen, T. Cooklev, C. Chen, and C. Pomalaza-Raez, *Modeling primary user emulation attacks and defenses in cognitive radio networks*, in *IPCCC*, 2009.

[58] T. Bansal, B. Chen, and P. Sinha, *FastProbe: Malicious user detection in cognitive radio networks through active transmissions*, in *INFOCOM*, 2014.

[59] A. Faggiani, E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, *Network sensing through smartphone-based crowdsourcing*, in *Proc. of SenSys*, 2013.

[60] Z. Zhang, L. Zhou, X. Zhao, G. Wang, Y. Su, M. Metzger, H. Zheng, and B. Y. Zhao, *On the validity of geosocial mobility traces*, in *Proc. of HotNets*, 2013.

[61] A. Gember, A. Akella, J. Pang, A. Varshavsky, and R. Caceres, *Obtaining in-context measurements of cellular network performance*, in *Proc. of Internet Measurement Conference*, 2012.

[62] S. Sen, J. Yoon, J. Hare, J. Ormont, and S. Banerjee, *Can they hear me now?: A case for a client-assisted approach to monitoring wide-area wireless networks*, in *Proc. of Internet Measurement Conference*, 2011.

[63] J. Shi, Z. Guan, C. Qiao, T. Melodia, D. Koutsonikolas, and G. Challen, *Crowdsourcing access network spectrum allocation using smartphones*, in *Proc. of HotNets*, 2014.

[64] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, *Zee: Zero-effort crowdsourcing for indoor localization*, in *MobiCom*, 2012.

[65] L. Li *et. al.*, *Experiencing and handling the diversity in data density and environmental locality in an indoor positioning service*, in *MobiCom*, 2014.

[66] L. Littman and B. Revare, *New times, new methods: Upgrading spectrum enforcement*, *Silicon Flatirons Center* (2014).

[67] `http://www.cnet.com/news/verizon-t-mobile-att-sprint-who-is-the-fastest-carrier-in-the-nation/`.

[68] S. Sen *et. al.*, *Can they hear me now?: a case for a client-assisted approach to monitoring wide-area wireless networks*, in *Proc. of Internet Measurement Conference*, 2011.

[69] A. Gember *et. al.*, *Obtaining in-context measurements of cellular network performance*, in *Proc. of IMC*, 2012.

[70] A. Achtzehn *et. al.*, *Crowdrem: Harnessing the power of the mobile crowd for flexible wireless network monitoring*, in *Proc. of HotMobile*, 2015.

[71] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, *Zee: zero-effort crowdsourcing for indoor localization*, in *MobiCom*, 2012.

[72] `http://www.opencellid.org/`.

[73] `http://openbmap.org/`.

[74] D. Pfammatter, D. Giustiniano, and V. Lenders, *A software-defined sensor architecture for large-scale wideband spectrum monitoring*, in *Proc. of IPSN*, 2015.

[75] `http://www.skyhookwireless.com/`.

[76] `http://www.antennasearch.com/`.

[77] `http://opensignal.com/`.

[78] `http://www.rootmetrics.com/`.

[79] `http://www.netradar.org/`.

[80] `http://googlemobile.blogspot.com/2008/06/google-enables-location-aware.html`.

[81] `http://jonspinney.com/thoughts/2008/3/24/how-does-googles-mylocation-really-work.html`.

[82] `http://franciscokattan.com/2010/02/06/dynamic-cell-id-clever-way-to-block-google-but-will-it-backfire/`.

[83] `https://developers.google.com/maps/documentation/geolocation/`.

[84] S. Shekhar and S. Chawla, *Spatial databases: a tour. Introduction to Spatial Data Mining (Chapter 7).* 2003.

[85] N. Megiddo, *Linear-time algorithms for linear programming in r^3 and related problems*, *SIAM journal on computing* (1983).

[86] A. Varshavsky, D. Pankratov, J. Krumm, and E. Lara, *Calibree: Calibration-free localization using relative distance estimations*, in *Proc. of Pervasive*, 2008.

[87] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, *Zero-configuration, robust indoor localization: Theory and experimentation*, in *INFOCOM*, 2006.

[88] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, *Indoor localization without the pain*, in *MobiCom*, 2010.

[89] P. Nurmi, S. Bhattacharya, and J. Kukkonen, *A grid-based algorithm for on-device gsm positioning*, in *Proc. of UbiComp*, 2010.

[90] J. Yang *et. al.*, *Accuracy characterization of cell tower localization*, in *Proc. of Ubicomp*, 2010.

[91] D. Cox and P. Lewis, *The statistical analysis of series of events*, .

[92] R. S. Yuill, *The standard deviational ellipse; an updated tool for spatial description*, *Geografiska Annaler* (1971).

[93] D. A. Griffith, *Spatial autocorrelation and spatial filtering: gaining understanding through theory and scientific visualization.* 2013.

[94] M. A. Hall, *Correlation-based feature selection for machine learning.* PhD thesis, University of Waikato, 1999.

[95] M. Hall *et. al.*, *The weka data mining software: an update*, *SIGKDD explorations* (2009).

[96] S. C. Johnson, *Hierarchical clustering schemes*, *Psychometrika* (1967).

[97] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, *SIAM Journal on Scientific Computing* (1998).

[98] H. Chernoff and E. Lehmann, *The use of maximum likelihood estimates in χ^2 tests for goodness-to-fit*, *The Annals of Mathematical Statistics* (1954).

[99] *Burglars Confess: Why Your Home is a Target*, 2013. `http://www.alarm.org/HomeSafety/BurglarsSpillAboutSecuritySystems.aspx`.

[100] *Minimize Blind Spots*, 2016. `http://www.vivint.com/neighborhood/tech-neighbor/minimize-blind-spots-for-your-security-camera-system/`.

[101] *Security Camera Blind Spots: How to Find and Avoid Them*, 2016. `https://reolink.com/find-and-avoid-security-camera-blind-spots/`.

[102] M. Kim, J. J. Fielding, and D. Kotz, *Risks of using ap locations discovered through war driving*, in *Proc. of Pervasive*, 2006.

[103] A. P. Subramanian, P. Deshpande, J. Gaojgao, and S. R. Das, *Drive-by localization of roadside wifi networks*, in *Proc. of INFOCOM*, 2008.

[104] M. Y. Chen *et. al.*, *Practical metropolitan-scale positioning for gsm phones*, in *Proc. of Ubicomp*, 2006.

[105] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, *Walkie-markie: indoor pathway mapping made easy*, in *Proc. of NSDI*, 2013.

[106] Z. Yang, C. Wu, and Y. Liu, *Locating in fingerprint space: wireless indoor localization with little human intervention*, in *MobiCom*, 2012.

[107] E. Neidhardt, A. Uzun, U. Bareth, and A. Kupper, *Estimating locations and coverage areas of mobile network cells based on crowdsourced data*, in *Proc. of WMNC*, 2013.

[108] M. Ulm, P. Widhalm, and N. Brandle, *Characterization of mobile phone localization errors with opencellid data*, in *Proc. of ICALT*, 2015.

[109] J. A. N. Rusvik, *Localizing cell towers from crowdsourced measurements*, Master's thesis, 2015.

[110] M. K. Marina, V. Radu, and K. Balampekos, *Impact of indoor-outdoor context on crowdsourcing based mobile coverage analysis*, in *Proceedings of Workshop on All Things Cellular: Operations, Applications and Challenges*, 2015.

[111] M. Molinari, M.-R. Fida, M. K. Marina, and A. Pescape, *Spatial interpolation based cellular coverage prediction with crowdsourced measurements*, in *Proc. of SIGCOMM Workshop on Crowdsourcing and Crowdsharing of Big (Internet) Data*, 2015.

[112] C. Meng *et. al.*, *Truth discovery on crowd sensing of correlated entities*, in *Proc. of SenSys*, 2015.

[113] L. Yi, B. Liu, and X. Li, *Eliminating noisy information in web pages for data mining*, in *Proc. of kdd*, 2003.

[114] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar, *Enhancing data analysis with noise removal*, *IEEE Transactions on Knowledge and Data Engineering* (2006).

[115] M. Soltanolkotabi and E. J. Candes, *A geometric analysis of subspace clustering with outliers*, *The Annals of Statistics* (2012).

[116] *Network Functions Virtualisation*, 2012.
`https://portal.etsi.org/NFV/NFV_White_Paper.pdf`.

[117] D. Collins *et. al.*, *Carrier grade voice over IP*, vol. 2. McGraw-Hill New York, 2001.

[118] S. Zhang *et. al.*, *Prefix: Switch failure prediction in datacenter networks*, in *Proc. of Sigmetrics*, 2018.

[119] A. W. Moore and D. Zuev, *Internet traffic classification using bayesian analysis techniques*, in *Proc. of Sigmetrics*, 2005.

[120] M. Sabhnani and G. Serpen, *Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context.*, in *MLMTA*, pp. 209–215, 2003.

[121] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* (1997).

[122] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, *et. al.*, *Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action*, in *SDN and OpenFlow World Congress*, 2012.

[123] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, *Detecting large-scale system problems by mining console logs*, in *Proc. of SOSP*, 2009.

[124] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, *What happened in my network: mining network events from router syslogs*, in *Proc. of Internet Measurement Conference*, 2010.

[125] M. Du, F. Li, G. Zheng, and V. Srikumar, *Deeplog: Anomaly detection and diagnosis from system logs through deep learning*, in *Proc. of CCS*, 2017.

[126] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, *LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems*, *arXiv preprint arXiv:1611.01726* (2016).

[127] M. Steinbach, G. Karypis, V. Kumar, *et. al.*, *A comparison of document clustering techniques*, in *KDD workshop on text mining*, vol. 400, pp. 525–526, 2000.

[128] D. Tang, B. Qin, and T. Liu, *Document modeling with gated recurrent neural network for sentiment classification*, in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015.

[129] J. Davis and M. Goadrich, *The relationship between precision-recall and roc curves*, in *Proc. of ICML*, ACM, 2006.

[130] N. V. Chawla, *Data mining for imbalanced datasets: An overview*, in *Data mining and knowledge discovery handbook*, pp. 875–886. Springer, 2009.

[131] A. P. Iyer, L. E. Li, and I. Stoica, *Automating diagnosis of cellular radio access network problems*, in *MobiCom*, 2017.

[132] J. A. Hartigan and M. A. Wong, *Algorithm as 136: A k-means clustering algorithm*, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **28** (1979), no. 1 100–108.

[133] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, *Self-taught learning: transfer learning from unlabeled data*, in *Proc. of ICML*, 2007.

[134] *Keras*, 2018. `https://keras.io/`.

[135] *Tensorflow*, 2018. `https://www.tensorflow.org/`.

[136] R. E. Hoskisson, M. A. Hitt, R. A. Johnson, and D. D. Moesel, *Construct validity of an objective (entropy) categorical measure of diversification strategy*, *Strategic management journal* **14** (1993), no. 3 215–235.

[137] D. M. Powers, *Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation*, .

[138] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, *Binary coding of speech spectrograms using a deep auto-encoder*, in *Proc. of INTERSPEECH*, 2010.

[139] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, *Automated it system failure prediction: A deep learning approach*, in *Proc. of Big Data*, 2016.

[140] Y. Wang, J. Wong, and A. Miner, *Anomaly intrusion detection using one class svm*, in *Proc. of IEEE SMC Information Assurance Workshop*, 2004.

[141] R. Sathyanarayanan, *Reliablity, resiliency and fault management in network function virtualization*, *ICN* (2016) 102.

[142] N. F. V. ETSI, "Resiliency requirements, etsi gs nfcv-rel 001, v1. 1.1."

[143] D. Kushnir and M. Goldstein, *Causality inference for failures in NFV*, in *Proc. of INFOCOM Workshops*, 2016.

[144] T. Niwa, M. Miyazawa, M. Hayashi, and R. Stadler, *Universal fault detection for NFV using som-based clustering*, in *Proc. of APNOMS*, 2015.

[145] M. Miyazawa, M. Hayashi, and R. Stadler, *vnmf: Distributed fault detection using clustering approach for network function virtualization*, in *Proc. of INM*, 2015.

[146] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun, *Anomaly detection and root cause localization in virtual network functions*, in *Proc. of ISSRE*, 2016.

[147] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, *Opprentice: Towards practical and automatic anomaly detection through machine learning*, in *Proc. of Internet Measurement Conference*, 2015.

[148] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, *Bluegene/l failure analysis and prediction models*, in *Proc. of DSN*, 2006.

[149] F. Salfner and M. Malek, *Using hidden semi-markov models for effective online failure prediction*, in *Proc. of SRDS*, 2007.

[150] I. Fronza, A. Sillitti, G. Succi, M. Terho, and J. Vlasenko, *Failure prediction based on log files using random indexing and support vector machines*, *Journal of Systems and Software* **86** (2013), no. 1 2–11.

[151] Viavi, "Interference hunting is fundamental to quality service." RCR Wireless News, Nov., 2016.
`https://www.rcrwireless.com/20161101/network-infrastructure/interference-hunting-fundamental-quality-service`.

[152] J. Ke, "Shanghai wants law on radio spectrum." Shine.cn, March, 2018.
`https://www.shine.cn/news/metro/1803061282/`.

[153] M. Amirijoo *et. al.*, *Cell outage management in LTE networks*, in *Proc. of ISWCS*, 2009.

[154] P. Denisowski, *Recognizing and resolving LTE/CATV interference issues*, *White Paper, Rohde and Schwarz* (2011).

[155] A. Nika *et. al.*, *Empirical validation of commodity spectrum monitoring*, in *Proc. of SenSys*, 2016.

[156] A. Chakraborty, U. Gupta, and S. R. Das, *Benchmarking resource usage for spectrum sensing on commodity mobile devices*, in *Proc. of HotWireless*, 2016.

[157] T. J. O'Shea, T. C. Clancy, and R. W. McGwier, *Recurrent neural radio anomaly detection*, *arXiv preprint arXiv:1611.00301* (2016).

[158] Q. Feng *et. al.*, *Anomaly detection of spectrum in wireless communication via deep auto-encoders*, *The Journal of Supercomputing* (2017).

[159] K. Tan, K. Zeng, D. Wu, and P. Mohapatra, *Detecting spectrum misuse in wireless networks*, in *Proc. of MASS*, 2012.

[160] R. Chen, J.-M. Park, and J. H. Reed, *Defense against primary user emulation attacks in cognitive radio networks*, *IEEE JSAC* **26** (2008), no. 1 25–37.

[161] P. Kaligineedi, M. Khabbazian, and V. K. Bhargava, *Malicious user detection in a cognitive radio cooperative sensing system*, *IEEE Trans. on Wireless Comm.* **9** (2010), no. 8 2488–2497.

[162] S. Liu, Y. Chen, W. Trappe, and L. J. Greenstein, *Aldo: An anomaly detection framework for dynamic spectrum access networks*, in *INFOCOM*, 2009.

[163] Z. Chen, T. Cooklev, C. Chen, and C. Pomalaza-Ráez, *Modeling primary user emulation attacks and defenses in cognitive radio networks*, in *Proc. of IPCCC*, 2009.

[164] L. Zhang, G. Ding, Q. Wu, and Z. Han, *Spectrum sensing under spectrum misuse behaviors: A multi-hypothesis test perspective*, *IEEE Trans. on Information Forensics and Security* **13** (2018), no. 4.

[165] R. K. Mehra and J. Peschon, *An innovations approach to fault detection and diagnosis in dynamic systems*, *Automatica* **7** (1971), no. 5 637–640.

[166] W.-L. Chin *et. al.*, *Channel-based detection of primary user emulation attacks in cognitive radios*, in *Proc. of VTC*, 2012.

[167] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural Computation* **9** (1997), no. 8 1735–1780.

[168] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, *Nature* **521** (2015), no. 7553 436.

[169] T. Guo *et. al.*, *Robust online time series prediction with recurrent neural networks*, in *Proc. of DSAA*, 2016.

[170] J. T. Connor, R. D. Martin, and L. E. Atlas, *Recurrent neural networks and robust time series prediction*, *IEEE Trans. on Neural Net.* **5** (1994), no. 2 240–254.

[171] T. Arici, S. Dikbas, and Y. Altunbasak, *A histogram modification framework and its application for image contrast enhancement*, *Trans. on Image Processing* **18** (2009), no. 9 1921–1935.

[172] Q. Cao, N. Balasubramanian, and A. Balasubramanian, *MobiRNN: Efficient recurrent neural network execution on mobile GPU*, in *Proc. of EMDL*, 2017.

[173] I. McGraw *et. al.*, *Personalized speech recognition on mobile devices*, in *Proc. of ICASSP*, 2016.

[174] S. J. Pan and Q. Yang, *A survey on transfer learning*, *Trans. on knowledge and data engineering* (2010).

[175] A. Koushik, E. Bentley, F. Hu, and S. Kumar, *A hardware testbed for learning-based spectrum handoff in cognitive radio networks*, *Journal of Network and Computer Applications* **106** (2018).

[176] R. Li *et. al.*, *Tact: A transfer actor-critic learning framework for energy saving in cellular radio access networks*, *Trans. on Wireless Communications* **13** (2014).

[177] E. Bastug, M. Bennis, and M. Debbah, *Anticipatory caching in small cell networks: A transfer learning approach*, in *Proc. of WAN*, 2014.

[178] G. F. Ciocarlie, U. Lindqvist, S. Nováczki, and H. Sanneck, *Detecting anomalies in cellular networks using an ensemble method*, in *Proc. of CNSM*, 2013.

[179] A. Bouillard, A. Junier, and B. Ronot, *Hidden anomaly detection in telecommunication networks*, in *Proc. of CNSM*, 2012.

[180] S. Nováczki, *An improved anomaly detection and diagnosis framework for mobile network operators*, in *Proc. of DRCN*, 2013.

[181] V. K. Gurbani *et. al.*, *Detecting and predicting outages in mobile networks with log data*, in *Proc. of ICC*, 2017.

[182] L. Qiu, P. Bahl, A. Rao, and L. Zhou, *Troubleshooting wireless mesh networks*, in *Proc. of SIGCOMM*, 2006.

[183] R. Chandra, V. N. Padmanabhan, and M. Zhang, *Wifiprofiler: cooperative diagnosis in wireless lans*, in *Proc. of MobiSys*, 2006.

[184] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker, *Mojo: A distributed physical layer anomaly detection system for 802.11 wlans*, in *Proc. of MobiSys*, 2006.

[185] Y.-C. Cheng *et. al.*, *Automating cross-layer diagnosis of enterprise wireless networks*, in *Proc. of SIGCOMM*, 2007.

[186] S. Liu, L. J. Greenstein, W. Trappe, and Y. Chen, *Detecting anomalous spectrum usage in dynamic spectrum access networks*, *Ad Hoc Networks* **10** (2012), no. 5 831–844.

[187] A. W. Min, K.-H. Kim, and K. G. Shin, *Robust cooperative sensing via state estimation in cognitive radio networks*, in *Proc. of DySPAN*, 2011.

[188] T. Bansal, B. Chen, and P. Sinha, *Fastprobe: Malicious user detection in cognitive radio networks through active transmissions*, in *INFOCOM*, 2014.

[189] D. Pu, Y. Shi, A. V. Ilyashenko, and A. M. Wyglinski, *Detecting primary user emulation attack in cognitive radio networks*, in *Proc. of GLOBECOM*, 2011.

[190] X. Xie and W. Wang, *Detecting primary user emulation attacks in cognitive radio networks via physical layer network coding*, *Procedia Computer Science* **21** (2013) 430–435.

[191] Y. Liu, P. Ning, and H. Dai, *Authenticating primary users' signals in cognitive radio networks via integrated cryptographic and wireless link signatures*, in *Proc. of S&P*, 2010.

[192] S. Rajendran *et. al.*, *Distributed deep learning models for wireless signal classification with low-cost spectrum sensors*, *arXiv preprint arXiv:1707.08908* (2017).

[193] A. Selim *et. al.*, *Spectrum monitoring for radar bands using deep convolutional neural networks*, *arXiv preprint arXiv:1705.00462* (2017).

[194] H. Wei, Y. Jia, and L. Wang, *Spectrum anomalies autonomous detection in cognitive radio using hidden markov models*, in *Proc. of IAEAC*, 2015.

[195] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, *Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks*, *arXiv preprint arXiv:1706.00527* (2017).

[196] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in *Proc. of NIPS*, 2012.

[197] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, in *Proc. of NIPS*, 2014.

[198] R. Salakhutdinov, J. Tenenbaum, and A. Torralba, *One-shot learning with a hierarchical nonparametric bayesian model*, in *Proc. of ICML Workshop on Unsupervised and Transfer Learning*, 2012.

[199] J. Snell, K. Swersky, and R. Zemel, *Prototypical networks for few-shot learning*, in *Proc. of NIPS*, 2017.

[200] R. Vilalta and Y. Drissi, *A perspective view and survey of meta-learning*, *Artificial intelligence review* **18** (2002), no. 2 77–95.

[201] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki, *Self-supervised learning of motion capture*, in *Proc. of NIPS*, 2017.

[202] R. S. Sutton, A. G. Barto, *et. al.*, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[203] N. Jay, N. H. Rotman, P. Godfrey, M. Schapira, and A. Tamar, *Internet congestion control via deep reinforcement learning*, *arXiv preprint arXiv:1810.03259* (2018).

[204] C. Dwork, *Differential privacy*, *Encyclopedia of Cryptography and Security* (2011) 338–340.

[205] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, *Model compression*, in *Proc. of KDD*, 2006.

[206] S. A. Osia, A. S. Shamsabadi, A. Taheri, K. Katevas, S. Sajadmanesh, H. R. Rabiee, N. D. Lane, and H. Haddadi, *A hybrid deep learning architecture for privacy-preserving mobile analytics*, *arXiv preprint arXiv:1703.02952* (2017).

[207] J. Frankle and M. Carbin, *The lottery ticket hypothesis: Finding sparse, trainable neural networks*, *arXiv preprint arXiv:1803.03635* (2018).

[208] A. Kurakin, I. Goodfellow, and S. Bengio, *Adversarial machine learning at scale*, in *Proc. of ICLR*, 2017.