



THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e. g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Deep Generative Models for Network Data Synthesis and Monitoring

Chuanhao Sun

Doctor of Philosophy
Institute of Computing System Architecture
School of Informatics
University of Edinburgh
2023

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. Part of the material used for the contributions made by my thesis has been published in the papers listed below. For the SPOTLIGHT work, the contribution of my co-authors, other than my supervisors, is limited to helping with the implementation of the respective software and data collection.

- **Chuanhao Sun, Kai Xu, Marco Fiore, Mahesh K. Marina, Yue Wang and Cezary Ziemlicki**, "AppShot: A Conditional Deep Generative Model for Synthesizing Service-Level Mobile Traffic Snapshots at City Scale," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4136-4150, Dec. 2022, doi: 10.1109/TNSM.2022.3199458.
- **Chuanhao Sun, Kai Xu, Mahesh K. Marina, and Howard Benn.** 2022. GenDT: mobile network drive testing made efficient with generative modeling. In Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22). Association for Computing Machinery, New York, NY, USA, 43–58.

(Chuanhao Sun)

To my parents

Abstract

Measurement and monitoring are fundamental tasks in all networks, enabling the downstream management and optimization of the network. Although networks inherently have abundant amounts of monitoring data, its access and effective measurement is another story. The challenges exist in many aspects. First, the inaccessibility of network monitoring data for external users, and it is hard to provide a high-fidelity dataset without leaking commercial sensitive information. Second, it could be very expensive to carry out effective data collection to cover a large-scale network system, considering the size of network growing, *i.e.*, cell number of radio network and the number of flows in the Internet Service Provider (ISP) network. Third, it is difficult to ensure fidelity and efficiency simultaneously in network monitoring, as the available resources in the network element that can be applied to support the measurement function are too limited to implement sophisticated mechanisms. Finally, understanding and explaining the behavior of the network becomes challenging due to its size and complex structure. Various emerging optimization-based solutions (*e.g.*, compressive sensing) or data-driven solutions (*e.g.* deep learning) have been proposed for the aforementioned challenges. However, the fidelity and efficiency of existing methods cannot yet meet the current network requirements.

The contributions made in this thesis significantly advance the state of the art in the domain of network measurement and monitoring techniques. Overall, we leverage cutting-edge machine learning technology, deep generative modeling, throughout the entire thesis. First, we design and realize APPSHOT, an efficient city-scale network traffic sharing with a conditional generative model, which only requires open-source contextual data during inference (*e.g.*, land use information and population distribution). Second, we develop an efficient drive testing system — GENDT, based on generative model, which combines graph neural networks, conditional generation, and quantified model uncertainty to enhance the efficiency of mobile drive testing. Third, we design and implement DISTILGAN, a high-fidelity, efficient, versatile, and real-time network telemetry system with latent GANs and spectral-temporal networks. Finally, we propose SPOTLIGHT, an accurate, explainable, and efficient anomaly detection system of the Open RAN (Radio Access Network) system. The lessons learned through this research are summarized, and interesting topics are discussed for future work in this domain. All proposed solutions have been evaluated with real-world datasets and applied to support different applications in real systems.

Lay Summary

The emerging new applications and traffic leads to a rapid increase of the current network in terms of size and complexity. The demand of lower latency, higher throughput, and higher robustness network has motivated a transition towards the next generation of networks. Especially on the mobile network side, 5G has been successfully commercialized in many regions around the world, with $> 10\times$ improvement on speed. Many network elements have evolved to support more intelligent network applications, such as programmable network switches that support computation directly on the switch and enable intelligent routing and telemetry.

While bringing better user experience, the size and complexity of current network makes it hard to monitor and understand. Network monitoring is one of the most fundamental tasks in network management, which is the way to obtain the network performance and then optimize the performance accordingly. The monitoring becomes much more difficult in IP network because of the massive volume of traffic; it is even very challenging to collect all IP packets. When considering the actual implementation, it becomes even harder to monitor, for example, one cellular basestation in Open RAN system has more than 600 parameters just for radio and Kubernetes, and in each city there are thousands of basestations. Such massive data require data mining and advanced measurement techniques.

The goal of this thesis is to enhance the data sharing, measurement, and anomaly detection in the context of current networks, including ISP networks, radio networks, IoT systems, etc. For data sharing and access, we propose APPSHOT, to generate a city-scale traffic map with solely open source context data as input, enabling data sharing without commercial sensitive leakage. To better understand and measure radio networks with a reasonable overhead, we propose GENDT, a conditional generator driven by the graph neural network (GNN) that generates radio parameters from open source network context, such as the location of the basestation and land use information. GENDT also enhances the efficiency of measurement by guiding the user to perform field tests to cover the regions help model training most. We also improve the efficiency of the general network telemetry task with DISTILGAN, a fast inference generator that reconstructs network measurement data from compressed format in an adaptive manner. Finally, we design and implement SPOTLIGHT, an explainable anomaly detection system based on a deep generator and causal discovery, where the communication overhead is significantly reduced by using a deep generative model on the far edge to drop the data that are irrelevant to the anomalous events. All the designs presented in the thesis are accompanied by concrete evaluations with real data and implementation with significant gain to the state of the art.

Acknowledgements

Firstly, I extend my deepest thanks to my supervisor, Mahesh K. Marina. His unwavering support and insightful guidance have been the cornerstone of my PhD journey. Mahesh has been more than an academic guide; he taught me the art of conducting quality research and the importance of striving for excellence. His lessons on enjoying the process and finding balance in life have shaped me both professionally and personally. He truly embodies the essence of a mentor in every sense of the word.

I am also immensely grateful to my second supervisor, Dr. Kai Xu. His expertise in high-quality machine learning research and his thoughtful advice on PhD life and career planning have been invaluable. His deep involvement in many of my projects and his role as a co-author have greatly contributed to my development as a researcher.

My sincere appreciation goes to my thesis examiners, Dr. Nishanth Sastry and Dr. Siddharth N, for their time, effort, and constructive feedback. Their inputs made my thesis defense not just a formal evaluation but an intellectually stimulating experience. I also extend my gratitude to Dr. Luo Mai for his instructive comments during my annual reviews, which significantly enhanced the quality of my work.

The collaborative environment during my PhD has been one of its highlights, and for this, I am thankful to my amazing colleagues - Jon Larrea, Leyang Xue, Andrew Ferguson, Ujjwal Pawar, and Zhihang Yuan. Their camaraderie and cooperation have been invaluable. Additionally, I extend my thanks to the senior members of our lab, Tanya Shreedhar, Alejandro Blanco, Mohamed Kassem, and Rajkarn Singh, for their support and guidance.

Special thanks are due to external experts who provided invaluable insights and assistance. I am indebted to Bozidar Radunovic and Xenofon Foukas for their guidance from a system researcher and industry perspective. My gratitude extends to Gianni Antichi for his profound insights into networking research and to Marco Fiore for his guidance in mobile networking, ML applications, and access to vital resources at IMDEA.

To my parents, Xiangye and Yanhong, words cannot express the depth of my gratitude. Your unconditional love, unwavering support, and belief in me have been the pillars of my strength. The journey of starting and completing this PhD would have been insurmountable without you.

Lastly, this PhD journey was uniquely challenging due to the pandemic. The support and help I received during these unprecedented times were crucial in making this experience a successful one. I am eternally grateful to everyone who stood by me, offering their help, encouragement, and understanding.

This thesis is not just a reflection of my work but a tapestry woven with the support and contributions of many. To all, I am eternally grateful. Thank you.

Contents

Abstract	5
1 Introduction	17
1.1 Network Measurement and Monitoring	17
1.2 Motivation	17
1.2.1 Challenges in Network Measurement and Monitoring	17
1.2.2 Generative Modeling for Network Measurement and Monitoring	18
1.3 Thesis Contributions	20
1.3.1 Conditional generation of service level mobile traffic data . .	20
1.3.2 Efficient mobile drive testing based on generative modeling .	21
1.3.3 Generative modeling based efficient and versatile network telemetry	21
1.3.4 Explainable generative modeling driven anomaly detection for Open RAN system	22
1.4 Thesis Organization	22
2 Background	23
2.1 Network Measurement and Monitoring	23
2.2 Generative Models	24
2.3 Generative Modeling for Networking	25
2.4 5G Radio Access Networks (RANs)	27
2.4.1 5G RAN general architecture	27
2.4.2 Open RAN architecture	27
2.5 Related Work	29
2.5.1 Service Level Traffic Data Generation	29
2.5.2 Drive Testing Data and the Application of Deep Learning . . .	30
2.5.3 Network Telemetry	31
2.5.4 Anomaly Detection with Network Measurement Data	33
3 AppShot	35
3.1 Introduction	35
3.2 Mobile Traffic and Context Data	36
3.3 Analysis of Mobile Traffic Characteristics Across Services and Cities	38
3.4 APPSHOT	40
3.4.1 Problem Statement	40
3.4.2 Patch based Learning Methods	41
3.4.3 Detailed Model Design	43

3.5	Performance Evaluation	45
3.5.1	Fidelity Metrics	45
3.5.2	Baselines	46
3.6	Results	47
3.6.1	Fidelity and Generalization	48
3.6.2	Detailed Comparisons with CartaGenie and SpectraGAN . . .	51
3.6.3	Benefit from Other Design Choices and Parameter Tuning . .	53
3.7	Use Cases of APPSHOT	54
3.8	Discussion	55
3.8.1	Limitations of APPSHOT	55
3.8.2	Potential Extensions	56
3.9	Summary	56
4	GenDT	57
4.1	Introduction	57
4.2	Background on Device Side Measurement of Radio Networks	60
4.2.1	Representative Radio Network KPIs	60
4.2.2	Measurement and Context Data	61
4.2.3	Network Context: Cell Information	61
4.2.4	Environment Context	62
4.3	Analysis of Data Characteristics	63
4.4	GENDT	64
4.4.1	Problem Statement	64
4.4.2	Overview of Proposed Solution	65
4.4.3	Generator	67
4.4.4	RESGEN	67
4.4.5	Batch Training and Generation	68
4.4.6	Stochastic Layers	69
4.5	Evaluation Methodology	70
4.5.1	Metrics	71
4.5.2	Baselines	71
4.6	Evaluation Results	72
4.6.1	Fidelity and Generalization	73
4.6.2	Long and Complex Scenarios	74
4.7	Measurement Efficiency	76
4.7.1	Model Uncertainty	76
4.7.2	Uncertainty Driven Measurement	76
4.8	Downstream Use Cases	77
4.8.1	Mobile Service Quality of Experience (QoE) Prediction . . .	77
4.8.2	Analysis of Handovers	78
4.9	Discussion	79
4.9.1	Why generating point coverage is insufficient	79
4.9.2	Comparison with Virtual Drive Testing	79
4.9.3	Weather and Other Critical Contexts	80
4.10	Summary	80

5	DistilGAN	81
5.1	Introduction	81
5.2	Requirements for Network Telemetry Methods	82
5.2.1	Fidelity	82
5.2.2	Efficiency	82
5.2.3	Versatility	82
5.2.4	Real-time	82
5.3	Motivation	82
5.3.1	Limitations of existing approaches	83
5.4	DISTILGAN	86
5.4.1	Overview	86
5.4.2	Generator Model Design for High Fidelity Data Stream Re- construction	87
5.4.3	Efficiency through Sampling Rate Adaptation	91
5.4.4	Q-value as coarse recovery error estimation	94
5.4.5	Mutual Difference of Reconstructed Time Series	95
5.4.6	Design Choices & Optimizations for Versatility and Real-Time Inference	96
5.5	Evaluation Methodology	98
5.5.1	Datasets	98
5.5.2	Metrics	99
5.6	Evaluation	100
5.6.1	DISTILGAN achieves High Fidelity and Efficiency	100
5.6.2	DISTILGAN is Versatile	101
5.6.3	DISTILGAN is Real-Time	103
5.6.4	Ablation Study	105
5.7	Downstream Use Cases of DISTILGAN	106
5.7.1	ISP Network Microburst Detection	106
5.7.2	O-RAN Anomaly Detection	106
5.7.3	IoT Smart Metering Analysis	107
5.8	Summary	107
6	SpotLight	109
6.1	Introduction	109
6.2	System Overview	110
6.2.1	Open RAN Architecture	110
6.2.2	Key Challenges in ORAN Anomaly Detection	111
6.2.3	Explanation in ORAN Anomaly Detection	112
6.3	Overview of SPOTLIGHT	112
6.3.1	System Architecture	112
6.3.2	Detection Method Description	114
6.4	Data collection	119
6.5	Evaluation Methodology	119
6.5.1	Evaluation setup	119
6.5.2	Dataset Creation	120
6.5.3	Representative Anomalies	121
6.5.4	Baselines	123

6.6	Evaluation Results	124
6.6.1	Accuracy	124
6.6.2	Explainability	125
6.6.3	Results with Multiple Anomalies	126
6.6.4	Efficiency	127
6.7	Real-world evaluation	127
6.7.1	Case Studies	127
6.7.2	Operational model	128
6.8	Discussion	128
6.8.1	Discriminator is not used to detect anomaly	128
6.8.2	How to understand the explanations	129
6.8.3	Choice of Models	129
6.9	Summary	129
7	Conclusions and Future Work	131
7.1	Conclusions	131
7.1.1	APPSHOT: Conditional Deep Generative Model for Synthesizing Service-Level Mobile Traffic Snapshots at City Scale	131
7.1.2	GENDT: Mobile Network Drive Testing Made Efficient with Generative Modeling	132
7.1.3	DISTILGAN: High Fidelity, Efficient, Versatile and Real-Time Network Telemetry with Deep Generative Modeling	132
7.1.4	SPOTLIGHT: Accurate, Explainable and Efficient Anomaly Detection for Open RAN	133
7.2	Future Work	133
7.2.1	Network Data Generation	133
7.2.2	Network Telemetry	133
7.2.3	Anomaly detection	134
A	Work and Publications	135
A.1	Publication Related to APPSHOT	135
A.2	Publication Related to GENDT	135
B	Implementation Details and Extra Results	137
B.1	Data Analysis and Model Details of GENDT	137
B.1.1	Visualization of Environment Context Attributes	137
B.1.2	Details of Stochastic Layers	137
B.1.3	Hyper Parameters	138
B.2	Discussion on DoppelGANger	139
B.3	Additional Evaluation and Use cases	139
B.3.1	Need to Support Long Series Generation	139
B.3.2	Ablation Study	140
B.3.3	Further Use Cases	141
B.4	Configuration Details of DISTILGAN	142
B.4.1	Epoch Length	142
B.4.2	Memory Configuration	142
B.5	5G RAN and IoT Smart Metering	143

B.5.1	Periodical Sampling	143
B.5.2	Threshold-based Sampling	143
B.6	Details of Baselines	144
B.6.1	Evaluation on synthetic time series	145
B.6.2	ISP Network Scenario: KPI Breakdown during Adaptive Sam- pling	147
B.6.3	Spectrum Change During DISTILGAN Processing	147
B.6.4	Synthetic Data Evaluation and Visualization	147

Chapter 1

Introduction

1.1 Network Measurement and Monitoring

Telecommunication network deeply shapes the world by performing most of the data communication tasks, building connections between arbitrary points to enable rapid transmission of information. It is critical to understand and optimize network performance, and the fundamental task in network management is measurement and monitoring of the network. Network measurement refers to the process to obtain a snapshot of network state, where the result could be in different formats: multivariate time series [98], images [159, 178], events [54], etc. When doing measurement in a continuous manner, we call this monitoring, where the network snapshot is kept up-to-date. Monitoring provides all the data needed for network management and optimization, and hence is a fundamental part in any network system.

1.2 Motivation

1.2.1 Challenges in Network Measurement and Monitoring

There exist several challenges in network measurement and monitoring that are yet to be resolved. In this thesis, we focus on four critical challenges in the network measurement and monitoring domain.

Inaccessibility of network monitoring data. Access to network monitoring data for research and innovation is severely restricted due to commercial sensitivity concerns of the operators. For example, to obtain network traffic data for all users in a city, all operations should be carried out within the operator’s premises under the control of the local Data Privacy Officer (DPO) and in compliance with applicable regulations, according to GDPR regulations (General Data Protection Regulation) [3], or shared with few selected external parties with highly restrictive NDAs.

Network measurement could be expensive. Measurement of a network might include a large number of field tests to achieve sufficient coverage. For example, mobile network drive tests at city scale are costly to conduct and time consuming (taking a few months [160, 161]). Unfortunately, as far as we know, operators do not share their drive testing data, and the availability of third-party dataset is very limited.

High fidelity and efficient network monitoring is difficult. Typically, obtaining fine-grained network monitoring data with affordable overhead is challenging due to the high-throughput monitoring traffic and hardware resource limitations of network equipment [69, 71, 182]. The need for a real-time control loop makes monitoring even more challenging, as some methods introduce significant overhead on data processing [71].

Understanding and explaining network behavior is difficult. Even when the issue of accessing rich network monitoring data is resolved, there is still the challenge of understanding network behavior for troubleshooting and optimization. The high-dimensional and time varying nature of network monitoring data makes it impractical for human analysis. For example, a single base station in a mobile radio access network (RAN) can produce several hundreds of measurement variables to analyse [54].

The aforementioned four challenges may seem independent but need to be all resolved for effective network monitoring in real-world networked systems.

1.2.2 Generative Modeling for Network Measurement and Monitoring

This thesis explores the application of generative models in networking problems, where they offer illuminating insights into handling network data characterized by substantial inherent randomness. In scenarios where deterministic correlations are absent, as exemplified by systems that transition between states in a stochastic manner akin to a Markov process, conventional methods fail to efficiently learn and represent the system’s dynamics. The complexity of these systems often exceeds simple stochastic models, further complicating the learning process. Generative models, however, adeptly navigate this complexity by learning the conditional probability distributions underlying different network systems. This approach enables them to capture and represent the inherent uncertainty and randomness more effectively, showcasing the potential of generative models in providing nuanced understandings of network dynamics.

Generative modeling can help address the aforementioned challenges. Many typical problems in network measurement and monitoring share a similar form as research problems in the domain of generative methods in machine learning, as we discuss in detail in §2.3. Overall, generative models can have significantly better performance in terms of fidelity, efficiency, and robustness than conventional methods. Nevertheless, leveraging the power of generative model is far from straightforward, as elaborated below. The new challenges when applying generative modeling to networking problems fall into three aspects: (1) access to training data; (2) unique constraints in networking, in terms of memory, computation, bandwidth, etc.; (3) Distinct statistical characteristics compared with common time series and images in other area.

Access to training data. As a data-driven method, the generative model must be trained with network data. However, access to network data is generally limited, which blocks third-party research activities. Instead of sharing the original data, one alternative is to share synthetic data that have sufficient fidelity to drive ML research, while avoiding leaking confidential messages. One possible workflow could be to train the data generation model and then share only synthetic copies to support other data-

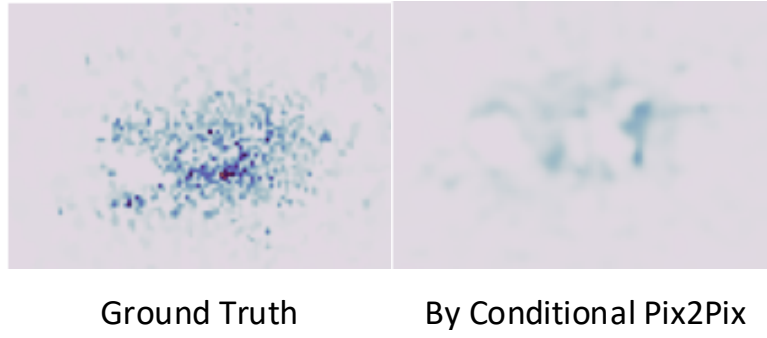


Figure 1.1: Classical generator performance on traffic map generation.

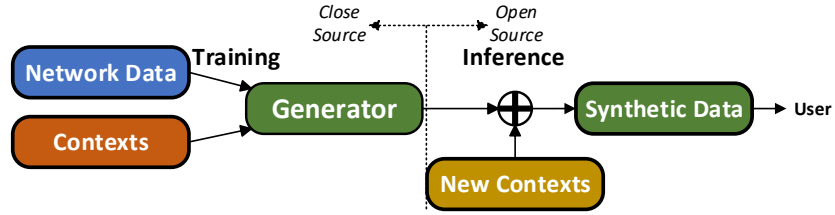


Figure 1.2: Workflow flow of generative modeling based network data sharing.

driven applications. As Figure 1.2 illustrates, the data owner (*e.g.*, network operator) can keep the trained generator as close source property but allow the user to upload the context of the target region. The owner will run the model inference and share the output with the user. The same commercial mode has been used in ChatGPT [1], etc.

Unique constraints in networks. Most ML algorithms are designed without considering system constraints. In certain tasks in networking, we require real-time processing of input data, which means the model inference should be finished before next input is ready; otherwise there will be queue waiting for inference. Even the inference is fast enough, there could still be issue on the communication bandwidth, where the model cannot work with fine-grain data (*e.g.*, SketchVisor [69] has to drop some IP packets as because the switch cannot parse all of them), sometimes strong noise will be applied (*e.g.*, radio interference). Those constraints require problem-specific design of an ML algorithm to enhance overall performance and robustness.

Different statistical characteristics. Can we directly use a conditional image generator to generate the traffic distribution from contextual information? Yes, but the performance is terrible. The feasibility is there because the traffic map has exactly the same data structure as the general image, running conditional GANs based on a classical structure such as Pixel2Pixel [80] would just give a blur output as Figure 1.1 shows. From a classical computer vision point of view, the synthetic traffic map still performs well in terms of few metrics, such as SSIM [27], because most of the place does not have significant traffic. However, in network management, we are more interested in those hot spots, which contribute most to the overall QoS, although it is only a few pixels on the image. To better accommodate the image generator into network data generation definitely requires designs that can better generate those hot spots. The consideration of network data characteristics is critical in all kinds of generative applications for networking.

1.3 Thesis Contributions

Overall, the primary contribution of this thesis lies not in advancing the core methodologies of machine learning (ML), but rather in the innovative application of state-of-the-art ML techniques to solve complex networking problems. This work demonstrates the effective integration of cutting-edge ML algorithms into the networking domain, showcasing how these sophisticated tools can be adapted and utilized in practical, real-world scenarios. Therefore, the significant value of this thesis resides in its applied ML focus, illustrating a novel and impactful approach to leveraging ML technologies in the field of networking.

In a more specific sense, the contribution of this thesis is primarily oriented toward addressing the challenges stated in §1.2. At a high level, we address the issue introduced by the ever-growing size of network with deep generative model, where those generative models can enhance the monitoring performance by generating missing data and making prediction for yet unseen region. When tackling the issue of implantation of generative models in networking, we first develop methods on synthesizing network data to support data-driven research, then design and implement a high-efficiency network telemetry system that considers the constraints in network elements. Throughout all problem solving, we accommodate the advances in generative modeling into networking application by introducing some special designs to work better with network data statistics, including preprocessing, neural network structure, and training process.

1.3.1 Conditional generation of service level mobile traffic data

The city-scale service level traffic data is rarely available to the public due to privacy concerns. Previous works [159] have shown that there is a significant correlation between traffic and context, where context refers to land use, population, time of day, etc. In CV area, there are existing works that transfer an image to another image with certain contexts. If we can represent those city contexts in a form of image, then we transfer the city-scale traffic generation task as a conditional image generation task.

Specifically, in this thread, we have the following technical contribution:

- Realize conditional generation of service level traffic map based on deep modified Unet model with noise input as latent information and adversarial loss in training process. We name this method APPSHOT.
- We introduce a batch-based generation technique for generation of arbitrary shape traffic maps.
- We fine-tune the sliding window mechanism in APPSHOT to achieve the best performance in terms of hot spots.
- We evaluate the proposed method with various metrics and downstreaming applications to demonstrate the fidelity of our generative model.
- We open source the synthetic dataset to support more data-driven research.

1.3.2 Efficient mobile drive testing based on generative modeling

In this thread, the focus is mainly on the experience of a single user. Conventional way to infer the experience of an end user includes (1) classic drive testing: field test in all scenarios and region; (2) virtual drive testing (VDT): doing test in lab environment to simulate real user experience; (3) minimal drive testing (MDT): real end user upload measurement data directly (crowd sourcing). The existing methods all have significant limitations. Classical Drive Testing has very low efficiency to cover a large area. VDT can only be used to perform certain equipment testing that can be simulated in the lab. MDT involves end-user data, which arises privacy concern, and the user might not volunteer for MDT. To address the limitations in existing drive testing methods, we propose GENDT, a user-side KPI (Key Performance Indicator) generator based on Graph Neural Network (GNN), and the uncertainty of generation will guide the measurement in the real world to cover more informative regions. The detail contribution is listed as follows.

- A GNN based generator that can aggregate various contextual data structure to generate high fidelity user side KPI time series.
- Using stochastic RNN to generate small-scale fading and a deep Gaussian-based method to generate large-scale fading in radio KPIs.
- An optimal measurement strategy driven by model uncertainty to reduce the actual field test needed to train GENDT.
- GENDT is evaluated together of various existing methods on multiple use cases to show the superiority of the proposed method.

1.3.3 Generative modeling based efficient and versatile network telemetry

The native computation capability of the network elements does not increase as fast as the network size, which overloads the memory and bandwidth of conventional network facilities. Although there is network hardware with advanced computation ability (*for example*, embedding GPUs in network switch), that hardware is expensive and hard to implement as a versatile solution. Therefore, in this thread, we propose a network telemetry system driven by generative modeling — DISTILGAN, which is adaptive to the state of the network. Moreover, DISTILGAN only requires a change in sampling rate or threshold on the network element side, all the computation involved is done in the cloud. The detailed contributions in DISTILGAN can be summarized as follows.

- A tailored spectral-temporal network based generator to reconstruct fine-grain telemetry data from samples.
- Optimization of inference process and highly parallel neural network structure to achieve $\sim 50\times$ gain on inference latency compared with compressive sensing based methods
- A heuristic mechanism to adapt to optimal sampling rate driven by estimation of reconstruction loss.

- Near-zero error measurement can be achieved with DISTILGAN with much less overhead than in previous works, which is also proven with different downstream tasks.

1.3.4 Explainable generative modeling driven anomaly detection for Open RAN system

The problem we face in network monitoring is not just about collection itself, but also about how we understand the network data. While in most cases we do not care about the network KPIs when it works perfectly, we pay more attention to the case where something anomalous happens, *a.k.a.*, anomaly detection. We develop and implement an anomaly detection system — SPOTLIGHT outperforms existing methods in the following aspects.

- High accuracy in anomaly detection with a generative model based on two-phase distribution learning.
- Using causal inference method and critical KPI filter to realize an accurate and readable anomaly detection report.
- Efficient implementation with the Open RAN system, lightweight computation at far-edge, and $4 \sim 7\times$ lower communication overhead than the state of the art.

1.4 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 introduces the background of the current network and the requirement of network monitoring. We also introduce related works in this chapter and list their limitations that motivate us to conduct research in this thesis.

Chapter 3 presents the detailed work on APPSHOT, generating high-fidelity city-scale traffic maps with open source contextual data as input.

Chapter 4 presents the work related to GENDT, starting from the measurement we made to train GENDT and the corresponding analysis of the network data that lead to the designs of GENDT. We also introduce how to use GENDT to reduce measurement overhead in the real world with use cases.

Chapter 5 investigates DISTILGAN, a data-driven network telemetry system, with details on how we improve the fidelity of telemetry, what techniques we applied to significantly reduce the inference latency and how DISTILGAN adapts to the state of the network. In this chapter, three scenarios are discussed to demonstrate the versatility of DISTILGAN, namely ISP network [25], 5G RAN, and IoT smart metering.

Chapter 6 carries out anomaly detection in the Open RAN system. In this chapter, we demonstrate the challenges in Open RAN anomaly detection, our system setup, and how we collect the measurement data. We then design and implement SPOTLIGHT, significantly outperforming the state of the art in terms of precision, explainability, and efficiency.

Chapter 7 concludes this thesis, summarizing the work presented and discussing the limitations of the contributions, as well as possible directions for future research.

Chapter 2

Background

2.1 Network Measurement and Monitoring

In this thesis, network refers to the **telecommunications network**, which is a group of nodes interconnected by telecommunications links that are used to exchange messages between nodes. Links may use a variety of technologies based on the methodologies of circuit switching, message switching, or packet switching, to pass messages and signals. Multiple nodes may cooperate to pass the message from an originating node to the destination node via multiple network hops. Examples of telecommunications networks include ISP (Internet service provider) network, RAN (radio access network), IoT (Internet of Things) network, etc.

Analyzing the performance of the network, identifying potential problems, and detecting malicious activities is critical. **Network Measurement** is the practice of taking a snapshot of the current state of the network in any format. **Network Monitoring** consists of the results of a series of measurements, or network monitoring can be defined as continuous network measurements. To understand the state and performance of the network, we need to collect the performance data of different network elements (*e.g.*, power, memory usage, etc.), as well as collect the information of the traffic sent within the network (*e.g.*, number of TCP flows, number of IP packets, etc.). Network monitoring provides the information that network administrators need to determine, in real time, whether a network is running optimally. Traffic engineering, quality of service, and anomaly detection also depend on monitoring for decision making.

In an ISP network or general IP network, the interest of measurement is mainly about the features of flows or aggregation of flows. While the definition of flow might vary between scenarios, the most common definition is two-tuples and five-tuples. By two-tuple — <Source IP, Destination IP>, it basically includes all traffic between two nodes in a specific direction, whereas the five-tuple:

<Source IP, Destination IP, Source Port Number, Destination Port Number, Transport Protocol>

further specify the port and protocol. For each flow or aggregation of flows, in general, we focus on the following typical measurements:

- Size of Flow (*i.e.*, bitrate).

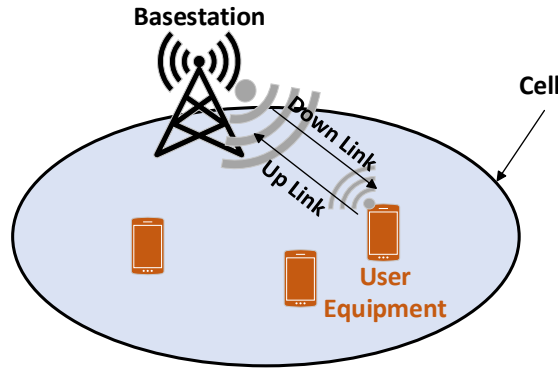


Figure 2.1: Major Components in a Mobile Network Cell

- Number of IP packet or special event, *for example*, number of SYN flags in the TCP flow.
- Change in flow size (change in bitrate).
- Distribution of flow size.
- Number of active flows.

In radio networks, there are many special parameters that appear mainly in wireless communication. The most representative radio network is the mobile network (*a.k.a.*, cellular network), which is a telecommunications network where the link to and from the end nodes is wireless and the network is distributed over land areas called cells, each served by at least one fixed location transceiver called basestation, as Figure 2.1 shows. In the context of mobile network measurement, we in general focus on the KPIs that reflect the features or performance of the wireless channel, for instance, SNR (Signal-to-Noise Ratio), CQI (Channel Quality Indicator), etc. While most of those special KPIs focus on the physical layer, the higher layer KPIs as stated in ISP network can still be applied to mobile network.

In addition to the ISP network and mobile network, there are many other different networks, such as sensor networks in an IoT system. Different systems might have totally different KPIs to be measured. The KPIs might cross different layers of the network or come from different elements of the system. In this thesis, we introduce the KPIs in each scenario before the evaluation.

2.2 Generative Models

Generative models have been widely deemed the most promising approaches to creative generation of real world data. From generating images [80, 156] to imitation of human conversation: ChatGPT [1], generative models keep updating people’s understanding of the power of artificial intelligence. The beauty of a generative model is that it generates a large set of data with significantly smaller amount of data on which we train them. In general, generative models have the potential to automatically learn the natural characteristics of a dataset, regardless of the categories or dimensions, even something entirely different.

Mathematically, a generative model is a statistical model of the joint probability distribution $P(X, Y)$ on a given observable variable X and the target variable Y or

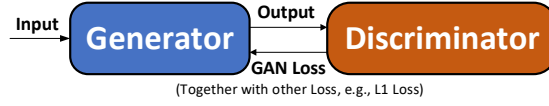


Figure 2.2: Schematic of adversarial training.

just $P(X)$ if there are no labels. Suppose the distribution of X and Y is $\mathbb{R}^X$ and $\mathbb{R}^Y$, the set of $\mathbb{R}^Y$ is generally much larger than $\mathbb{R}^X$. With the rise of deep learning, a new family of methods is formed through the combination of generative models and deep neural networks. The conventional generative model includes VAE (variational autoencoder) [88], Gaussian mixture model [146], etc.

Generative models become much more powerful after the introduction of adversarial training, which is known as GAN (Generative Adversarial Network) [73], a type of deep generative model. As its name indicates, adversarial training includes a pair of generative model (*a.k.a.*, generator) and discriminative model (*a.k.a.*, discriminator), and trains them together in an adversarial manner, as Figure 2.2 illustrates. The discriminator is a model of the conditional probability of the target Y , given an observation x , symbolically $P(Y|X = x)$. Originally introduced to study attacks on machine learning algorithms, adversarial training has been proven to be efficient in the performance of deep learning models.

As an extension of classical deep learning models, the generative model in general shares a similar neural network structure. The main difference is in the training process, the way to introduce latent information (*e.g.*, noise input), and the inference process (*e.g.*, enable variational inference). On one hand, these differences make the training and robustness of generative models a bit challenging, for instance, the model collapse issue of unconditional generative models. On the other hand, because of these new functions, GAN shows significant gain on distribution learning, which then contributes to better fidelity or accuracy performance.

2.3 Generative Modeling for Networking

Inspired by the characteristic of generative models that generate a much larger dataset with a relatively much smaller training set, the generative model can be applied to generate network data for various purposes. So far, in previous work [170, 178, 179], generative models have been proven to be powerful in network measurement and monitoring in many aspects, from data prediction to anomaly detection. Meanwhile, there is a trend to include native support for machine learning (ML) functions in various networks, making the application of generative models feasible.

Generative models have shown great power on creating content. From a pure data science point of view, network measurement data share a structure similar to that of general objects. For instance, the traffic snapshot of a whole city could be taken as an image. Starting from the data structure, many tasks in networking can be represented by more general tasks in machine learning. As the example in Figure 2.3 shows, the generative model that generates an image from text description could be applied to generate a traffic map on a city scale. By generating such a traffic map, we can share the traffic data without leaking privacy information. Also, during the implementation of a new network, we can utilize such technology to enable 'What if' functions by predicting traffic or coverage performance with available context information in target

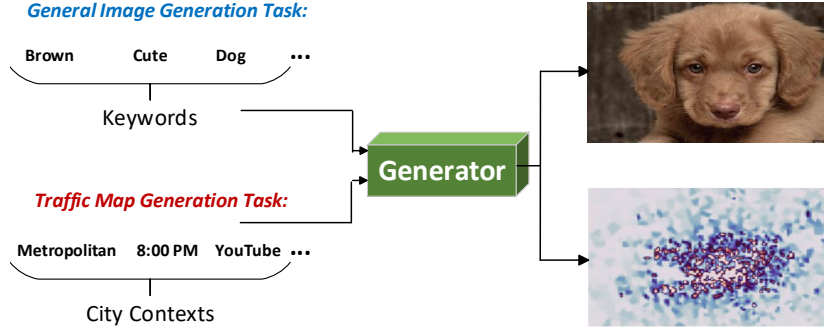


Figure 2.3: General Computer Vision Task and Network Traffic Generation.

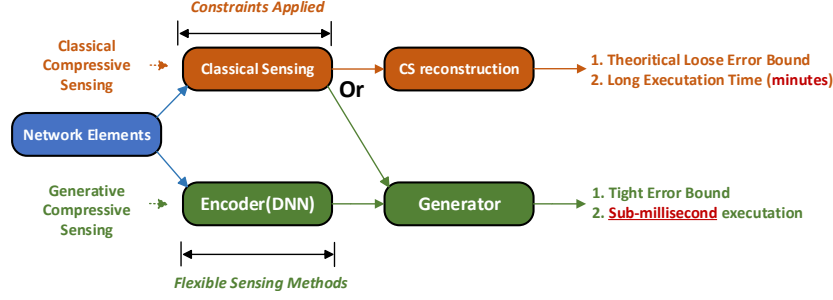


Figure 2.4: Classical Compressive Sensing and Generative model driven case.

region.

Generative model also brings advanced computing capability into the networking area. Optimization problem in networking, in general, includes heavy CPU computation that cannot be directly accelerated by GPUs [157]. The generation model can be implemented in one or several steps in the optimization problem solving process. One typical scenario is compressive sensing (CS). The process of a network measurement system based on CS is illustrated in Figure 2.4, where the measurement data is collected on network elements, then through the “sensing” process in CS algorithm, the data are compressed into a dense representation and hence save the bandwidth for the following transmission. Then on the receiver side, we run the CS reconstruction to recover the original data. In the classical CS, the sensing process has certain constraints; for instance, the data need to be sparse. The conventional reconstruction process in general includes an optimization problem solving process, which is slow when the dimension of input data is high. The optimization problem in CS can be taken as generate the original data based on the result after sensing, and then the optimization problem is resolved via model inference and can be done with milliseconds level latency or even less. In addition, the sensing process could be done with machine learning as well to obtain an optimal representation of data that matches the corresponding generator on the receiver side.

There are more use cases of the generative model in networking and we cannot list them all due to space. In general, we can find many use cases in networking that can be improved by generative models because they broadly share similar data structure and objective.

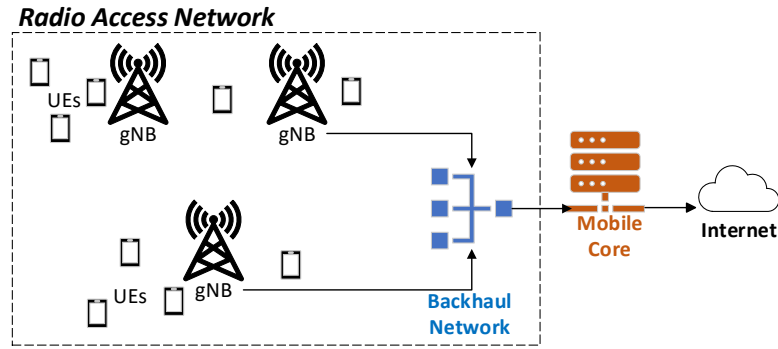


Figure 2.5: 5G Network Overview.

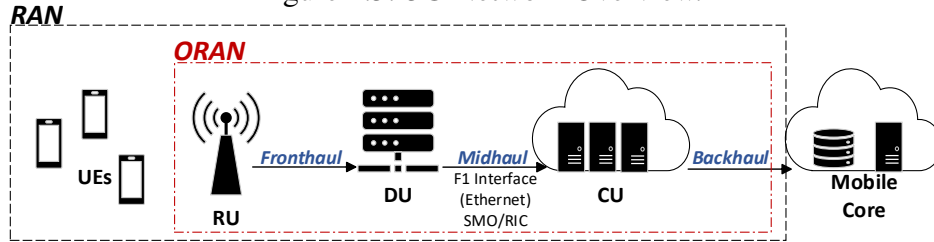


Figure 2.6: Open RAN Overview. RU: Remote Unit (radio communication), DU: Distributed Unit, CU: Central Unit, UE: User Equipment, SMO: Service Management and Orchestration.

2.4 5G Radio Access Networks (RANs)

2.4.1 5G RAN general architecture

Overall architecture of 5G RAN (Radio Access Network) would not look much different from the previous generation (4G or LTE). The definition of RAN is explained in Figure 2.5, including user equipment (UE), gNB (gNodeB, 5G basestation). There is backhaul connects RAN with mobile core network. The key difference between the 5G RAN from the previous generation is the different protocol and implementation. Among all of these differences, the most outstanding one would be that the gNB internal structure is split into two parts called CU (Central Unit) and DU (Distributed Unit), which will be illustrated in detail with a representative implementation of 5G RAN on open-source hardware — O-RAN (**O**pen **R**AN) in the following part of this section. For the monitoring of mobile core and Internet traffic, there is no structural change for 5G, as they are basically IP network systems, and hence we do not provide further discussion in this part.

2.4.2 Open RAN architecture

The detailed network elements of ORAN is illustrated in Figure 2.6. ORAN is not fully overlapped with the definition of RAN, as the user equipment is not included. ORAN consists of three main components, RU, DU, and CU. While RU is responsible for the radio communication part just as in the previous generation, the split CU/DU is the new function in 5G RAN. The separation of CU and DU helps virtualize network functionalities, which potentially contributes to flexibility and cost reduction. The CU and DU in the ORAN system is operated on general CPUs, making the implementation

of RAN easier.

As Figure 2.6 shows, RAN disaggregation splits base stations into different functional units, effectively embracing and extending the functional disaggregation paradigm proposed by 3GPP for Next-Generation Node Bases (gNB) of NR. The gNB is divided into a central unit (CU), a distributed unit (DU), and a Radio Unit (RU). The CU is further split into two logical components, one for the Control Plane (CP) and one for the User Plane (UP). This logical split allows different functionalities to be deployed at different locations of the network, as well as on different hardware platforms. For example, CUs and DUs can be virtualized on white-box servers at the edge (with hardware acceleration for some of the physical layer functionalities), while the RUs are generally implemented on field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) boards and deployed close to RF antennas.

The O-RAN Alliance has evaluated the different RU/DU split options proposed by the 3GPP, with specific interest in alternatives for physical layer split across the RU and the DU. The 7.2x version of RU/DU split strikes a balance between simplicity of the RU and the data rates and latency required on the interface between the RU and the DU. In split 7.2x, the RU performs time-domain functionalities, with precoding, Fast Fourier Transform (FFT), cyclic prefix addition/removal, and Radio Frequency (RF) operations, which makes the RU inexpensive and easy to deploy. The DU then takes care of the remaining functionalities of the physical layer and of the Medium Access Control (MAC) and Radio Link Control (RLC) layers, including scrambling, modulation, layer mapping, part of precoding, and mapping into physical resource blocks. The operations of these three layers are generally tightly synchronized, as the MAC layer generates Transport Blocks (TBs) for the physical layer using data buffered at the RLC layer. Finally, the CU units (CP and UP) implement the higher layers of the 3GPP stack, that is, the Radio Resource Control (RRC) layer, which manages the life cycle of the connection; the Service Data Adaptation Protocol (SDAP) layer, which manages the Quality of Service (QoS) of the traffic flows (also known as bearers); and the Packet Data Convergence Protocol (PDCP) layer, which takes care of reordering, packet duplication, and encryption for the air interface, among others.

Another innovation is represented by the RICs in Figure 2.6, which introduce programmable components that can run optimization routines with closed-loop control and orchestrate the RAN. The nonreal-time (or non-RT) RIC is a component of the Service Management and Orchestration (SMO) framework. The non-RT RIC provides guidance, enrichment information, and management of ML models for the near-RT RIC. Additionally, the non-RT RIC can influence SMO operations, which gives the non-RT RIC the ability to indirectly govern all components of the O-RAN architecture connected to the SMO, thus making decisions and applying policies that influence thousands of devices. The near real-time (or near-RT) RIC is deployed at the edge of the network and operates control loops with a periodicity between 10 ms and 1s. Near-RT RIC consists of multiple applications supporting custom logic, called xApps, and of the services that are required to support the execution of the xApps. An xApp is a microservice that can be used to manage radio resources through specific interfaces and service models.

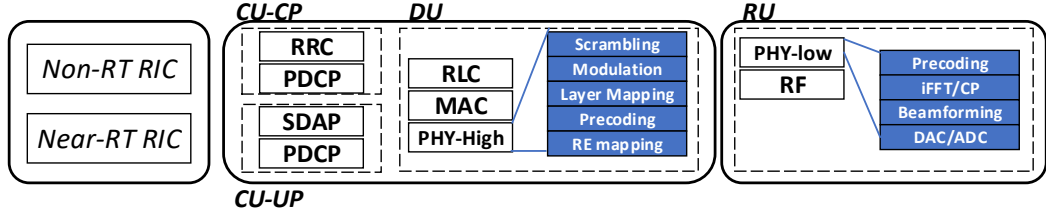


Figure 2.7: Open RAN Functions.

2.5 Related Work

2.5.1 Service Level Traffic Data Generation

Traditional network traffic generation focuses on creating different packet-level workloads. There are a number of tools that exist for this purpose (*e.g.*, iPerf, MGEN, Ostinato) and are also embedded in popular network simulators (*e.g.*, ns-3). Some of these tools like D-ITG [2, 22] support modeling different applications through parameterized probability distributions for packet sizes, their inter-arrival times, etc. This form of traffic generation does not have a spatial dimension. In contrast, our focus is on generating snapshots of application/service level mobile traffic volumes (aggregated across multiple users and flows) at different locations of a target region (*e.g.*, a city).

We are unaware of any prior work for generation of service-level mobile network traffic data. The few related works that exist in the mobile networking context [43, 107, 178, 180] focus on overall traffic across *all* services. Di Francesco *et al.* [43] propose an approach for assembling a cellular dataset for a given region by integrating multiple sources of data, including census data for population distribution, base station locations and estimation of data demand per subscriber. For the data demand, they simply model this as a probability distribution based on operator provided data on overall mobile traffic across all services and then sample from it. We consider this approach as a baseline in our evaluations and highlight its limitations in handling traffic correlations. In another mobile traffic related work, Bo *et al.* [107] target generation of mobile traffic patterns for a region focusing on hotspots through geotagged Twitter data for that region. Here again, only total traffic volume across all services is considered and not at the individual service level like we do. Moreover, access to Twitter data is no more easier than accessing mobile traffic data whereas we base our generation on context data for the target region that can be easily obtained from public sources.

SpectraGAN [180] and CartaGenie [178] are recent proposals that can be viewed as the state of the art on mobile traffic data generation. As in our work, SpectraGAN and CartaGenie take a conditional deep generative modeling approach but focus on generation of spatial or spatiotemporal data for *total* traffic volumes in a city. We target a different and orthogonal dimension, *i.e.*, on the *individual service level* contributions that make up the total traffic. As we show in our comparative evaluation, applying SpectraGAN or CartaGenie for our purpose yields poor quality generation due to its inability to model inter-service correlations and their relation to total traffic.

As we represent city-scale mobile traffic snapshots as images, their generation at service level can be viewed as a multi-channel image generation problem. Furthermore, since we aim at conditional generation using contextual attributes as a multi-channel image input, image translation works from the computer vision domain are

particularly relevant. Pix2Pix [80] is a representative prior work on conditional image translation. When applied to our problem setting, this work has several key limitations as we show in our evaluation: (i) it does not take particular care to capture correlations among channels (services in our problem); (ii) it fails to model variation in the data from using just dropout for stochasticity; (iii) it can also result in undesirable edge effects and artefacts when generating traffic maps for arbitrary sized regions. These limitations also apply to other related works from the computer vision literature (e.g., Style-GAN [60], Cycle-GAN [198]), which are essentially rooted in the fact that they do not cater to the unique requirements of mobile service traffic map generation. The works from the transportation domain, exemplified by Traffic-GAN [197], for road vehicle traffic generation are also broadly related. However, these works do not differentiate between different vehicle types (individual mobile services in our case) and also make a strong assumption of knowing correlations among traffic on different roads for the target region, which is unrealistic.

Besides generation of multi-service mobile traffic maps, our work also includes an analysis of mobile network traffic across different services and cities. This part is novel compared to prior service-oriented mobile traffic analysis works (e.g., [111, 159]) by focusing on the key characteristics that need to be kept in mind when generating service-level mobile traffic data. In particular, unlike [159], we analyze the correlation between traffic of different mobile services as well as with a wide range of contextual attributes beyond urbanization. Compared to [111], we study the similarities and differences in mobile traffic across cities, with a focus on peak periods, traffic stochasticity, hotspot density and distribution.

Takeaways. Overall, the conditional generative model in computer vision (CV) and related domain is very promising at service-level data generation. However, directly applying existing methods to network data generation cannot generate a high-fidelity traffic map due to the special characteristics of service level traffic. A tailored generator is needed to generate high-fidelity service level traffic.

2.5.2 Drive Testing Data and the Application of Deep Learning

In the context of mobile network drive testing and is aimed at reducing its cost associated with measurement data collection. As stated at the outset, the VDT approach [26, 122, 124] is limited to device/equipment testing and so is unsuitable for this purpose. The other alternative approaches involving user device based measurement collection via MDT [6, 84, 158] or crowd-sourcing [10, 51, 123, 125] are hindered by insufficient incentives and privacy concerns. To the best of our knowledge, our work is the first to explore the generative modeling approach towards making drive testing efficient and cost effective.

Broadly related are the works focusing on coverage mapping and pathloss prediction, which can be seen as a subset of drive testing use cases. In contrast to traditional methods including ray-tracing [138], recent work (e.g., [10, 51, 164, 176]) has adopted statistical and machine learning approaches for measurement or computational efficiency. Alimpertis et al. [10] propose a random forests based model for prediction of signal strength (RSRP) map, whereas Thrane et al. [164] present a convolutional neural network (CNN) based supervised spatial regression model that maps satellite images of a target region to signal quality parameters like RSRP and RSRQ in that region.

On the other hand, [176] focuses on pathloss prediction using multi-layer perceptron (MLP) based neural network model. The above mentioned works cannot mimic measurements with drive testing as they do not have a notion of user trajectory or temporal variations. They also make a simplifying but inaccurate assumption that serving cell at each location is fixed and known. Moreover, the model in [164] due to being trained with satellite images for a specific region does not generalize beyond that region. In contrast, our proposed GENDT approach overcomes the above limitations through a tailored and novel deep generative model.

GENDT leverages graph neural networks (GNNs) [13] to effectively handle varying network context around a drive testing trajectory. While there have been some recent works employing GNNs for time-series prediction problems (e.g., [104, 168]), to our knowledge, ours is the first work on GNN based time-series data ‘generation’. As noted in prior work [179], data generation is a much harder task than prediction. We comparatively evaluate our model with the LSTM-GNN model [168].

Using deep generative models, especially generative adversarial networks (GANs) and variational autoencoders (VAEs), for data synthesis is of prime interest currently [121]. Such models are being used to generate data for machine learning, in finance, healthcare and other domains. Within the mobile networking domain, there have been few recent works proposing deep generative models for various types of network and wireless data. The potential for GANs to generate physical layer channel response samples for MIMO channels has been discussed in [184]. SpectraGAN [179] is another broadly related work in this domain that targets the generation of spatiotemporal mobile traffic data. Unlike our setting, mobile traffic data has certain unique properties such as ‘recurring’ patterns that are exploited in SpectraGAN for effective data generation.

Works on multivariate time-series synthesis in general are related given our problem involves generating time-series data for multiple radio network KPIs. Existing work [32, 93, 98], however, targets very different problems from ours. For instance, in [93], an unconditional GAN based multivariate time-series synthesis model is introduced to generate data for resource utilization measurement of CDN caches whereas we target a conditional data generation problem. As another example, Chen et al. [32] focus on mitigating the severe class imbalance in the data for predicting rare events (e.g., solar flares).

Among these works, DoppelGANger (DG) [98] is a more closely related work that is aimed at unconditional GAN based generation of multivariate time-series data for networks and systems (e.g., Wikipedia article views over time, network monitoring data over time, resource usage in compute clusters).

Takeaways. To our knowledge, there is no generative model in the literature that is tailored to generate multivariate time series that reflect stochastic features and achieve high fidelity at the same time. Even with the generation method, we still need to design a method to reduce the overhead of field test, because the cost of collecting sufficient training data is high when conducting city-scale measurement.

2.5.3 Network Telemetry

Sampling. In some system when the memory and bandwidth is sufficient, Nyquist sampling can be applied as a lossless data compression method in real-time [186]. However, if we look at more general network telemetry task, there are two limitations

of Nyquist sampling: (1) the underlying system cannot achieve the Nyquist rate due to the limitation of hardware — sub-Nyquist is essential sometimes; (2) Nyquist rate changes a lot across different periods and hard to search the correct rate. Therefore, Nyquist sampling cannot guarantee the feasibility in general network measurement as equipment such as switches may have very limited memory, and the robustness of Nyquist sampling is also poor because it relies on heuristic methods [135] to determine the correct Nyquist rate and the searching process is hard to converge when the signal bandwidth changes very fast.

Sampling regardless Nyquist rate is also used in network telemetry [145, 166, 171]. In those scenario, sampling is applied to significantly reduce the memory or bandwidth overhead, and only an approximation of the ground truth measurement can be obtained. We therefore did not include these works in the evaluation as they cannot meet the high-fidelity requirement at suggested configuration.

Sketching [53] is the common method in network telemetry for abstracting, where approximation is introduced to record the network KPIs of interest. Because lack of data recovery design, conventional sketching methods [70, 95, 100, 101, 182] either focus on very high level statistics or ignore less significant part of network to meet the memory/bandwidth budget, for instance SketchVisor [69] directly drop some IP packets to make sure the sketch is executable with limited memory size. Those approximation leads to a poor fidelity on its result, SketchVisor [69] might have more than 30% flows measured with significant error under the recommended memory configuration as there still be significant IP packet dropping. Besides, those methods cannot adapt to the dynamics in the network and only works with the initial configuration, without the ability to report the measurement error, leaving the robustness in complex network environment vulnerable. In practice, people has to configure them with the theoretical upper bound error and network throughput, which is inconvenient and would cause resource waste.

Sketching Based Sensing. The discussion of combination of sketching and CS has started from early years, in [36, 92, 103] people notice that sketching could be taken as a sensing method in CS system. Recent years we there are few successful usages of classic CS in general sketching system, to further reduce the measurement amount on data plane or sensors [69, 71]. In conventional setup [69], because the features such as orthonormality of the sensing matrix is not fine tuned, the fidelity of recovery is limited. Method such as [71] fine tune the sensing matrix as well as the counter structure to achieve near-zero error recovery and even less memory consumption. Nevertheless, there are another two limitation of classic CS: (1) Recovery time is unacceptable long when the data dimension is high; (2) Estimate the sparsity of network data is challenging. The first limitation is more significant for large network measurement, when there is 100Ks of flows, it takes minutes for the optimization solver to converge. The second limitation means it is very difficult to know if we allocate sufficient sampling rate (or memory) because we do not have effective method to estimate the recovery error, and therefore poor robustness upon significant data sparsity changes.

Classical Compressive Sensing (CS). In a most general setup, classical CS highly rely on the sparsity of data. Network data show sparsity in diverse scenarios. [33, 90, 196] leverage compressive sensing to recover missing values in traffic matrices. [23] applies CS to network link tomography.

Deep Learning based Compressive Sensing (DL-CS). Besides sparsity, there are

latent structure and correlation between samples in real-world network data, such as periodicity and self-similarity [40, 82, 133], which can be used for data recovery beyond Nyquist rate. Classic CS relies on data sparsity without looking into other inherent structure on target data. DL-CS [62, 174, 177, 181] is then proposed to leverage the other structural information, and also in general has much lower recovery time (around two orders). While shows better recovery fidelity in variate domains — image processing [156, 195], wireless networks [65, 108], etc., there is no thorough investigation of DL-CS in network telemetry. Specially in network telemetry, the DL-CS with fancy encoder on sender side is not applicable because the sender side in general do not sufficient resources to process high data throughput with encoders. Broadly, we can take the latest time series generation and imputation models [49, 91, 102, 106, 108, 162, 188] as DL-CS methods without an encoder and designed for time series. However, the existing methods cannot give a high-fidelity reconstruction when the sampling rate is very low, and the recovery time is unacceptable with some latest models. The CS theory is also applied some lossless compression algorithm with deep learning [61, 109], we classify those methods to DL-CS family as well. While achieving higher compression ratio and much lower bandwidth consumption, the computation complexity is not affordable in most of network equipment, and they also need to be operated on large window size to achieve significant compression ratio — the actual memory and latency overhead is very high in measurement system.

Takeaways. Generally speaking, the existing methods make different trade-offs between the various requirements — **high reliability, efficiency, versatility and real-time**, and as such none of them meets all the requirements mentioned above. This observation motivates us to pursue a new and powerful approach to network telemetry that is better suited to meet all requirements.

2.5.4 Anomaly Detection with Network Measurement Data

Network measurement data in general comes in the form of multi-variate time series, where each variate corresponds to specific. Time series anomaly detection is an active area of research in the machine learning domain [19]. In line with what is noted above, the state-of-the-art methods are prediction based (e.g., GDN [41]), reconstruction based (e.g., MADGAN [94]) or combine both (e.g., TranAD [170], VAE-LSTM [97]). These methods have poor precision (high false alarms) when applied to our Open RAN setting. Moreover, with the exception of a few methods like GDN [41], most existing time series anomaly detection methods lack explainability.

From a method design perspective, the problem we target is essentially multivariate time series anomaly detection [19]. In the RAN context, prior work (e.g., [87]) has shown that commonly used non time series anomaly detection methods (e.g., Z-Score based, robust covariance, one-class SVM) [5, 89], and supervised binary classification based anomaly detection, as considered in early works (e.g., [81]), are ineffective. Consequently, state-of-the-art approaches for RAN anomaly detection broadly fall under two classes: (i) time series prediction with recurrent neural networks (e.g., LSTM) [31, 87, 169, 192]; (ii) reconstruction based with autoencoders [87, 113, 169]. Both these approaches are limited by the unwieldy challenge of having to determine a right threshold for prediction/reconstruction errors.

Explainability or root cause analysis has been considered in some prior works on

anomaly detection in traditional RANs [31, 143, 192]. A common approach is to augment an anomaly detection method with SHAP (SHapley Additive exPlanations) [105] or similar model-agnostic explainers, for identifying important features/KPIs responsible for the detection of anomalies [31, 192]. Interpretable shallow ML models such as decision trees have also been used [143]. Explainability of AI models is starting to be recognized as an important requirement in the Open RAN context [24]. However, we are unaware of any existing work on explainable anomaly detection for this context.

Takeaways. We do not find a method in the literature that jointly considers Accurate, Explainable and Efficient anomaly detection. In the ORAN system, the task is a bit tricky because only detecting the anomaly accurately is not sufficient, we have to understand the root cause from hundreds of parameters. Meanwhile, anomaly detection should be conducted in a lightweight way, with the consideration of the limited computation resource available on the basestation and transmission latency when the cloud is involved.

Chapter 3

AppShot

3.1 Introduction

In this chapter, we focus on the first problem studied in this thesis, conditional generation of service level mobile traffic data. As stated in §1.3.1, larger scale service-level mobile traffic data enables research studies and innovative applications in networking domain, with a potential to shape future service-oriented communication systems and beyond. However, real-world datasets reporting measurements at the individual service level are hard to access as such data is deemed commercially sensitive by operators. Such restriction can be relieved by a generative model that can generate high fidelity traffic map solely with open source contextual data. The generated traffic map is distinct to the real traffic map and hence the risk of leaking sensitive information would be much lower than manually cleaned dataset. Meanwhile, the fidelity of generation is sufficient to support data driven applications and studies.

Designing a data synthesis model that can generate high-fidelity service-level mobile traffic snapshots and generalizes well to new regions is challenging due to a number of reasons. First, the publicly available context data for a target region may not fully determine the mobile services traffic for that region and in general cannot capture the stochasticity inherent to mobile traffic. Second, mobile traffic is known to have complex spatiotemporal correlations both overall and at service level [134, 153], which need to be captured by the model. Third, locations and times with high traffic intensity (which we refer to as *hotspots* in this paper) are particularly important for downstream use cases on research management and beyond (e.g., [83]), and need to be faithfully modeled. Fourth, the model should correctly capture correlations between traffic for different services and their relative contribution to overall traffic. Finally, the model should be flexible in accommodating the fact that the target regions for traffic generation may differ widely in their geographical dimensions as well as contextual attributes and traffic characteristics. All those challenges motivate us to develop APPSHOT, which to best our knowledge the first high fidelity generative model for service-level city-scale traffic map.

In this chapter, we provide an in-depth discussion regarding the design of APPSHOT. The detailed neural network structure, training method, and other special designs are included. The high-fidelity of APPSHOT makes the generated data of APPSHOT an open-source real-world traffic dataset that helps various downstreaming ap-

plications.

This chapter is structured as follows. The next section elaborate mobile traffic and context data relevant to APPSHOT. Then in §3.3, we conduct an analysis of the aforementioned data, including service-level traffic characteristics and correlation between traffic and context. The proposed generative model APPSHOT is described in detail in §3.4. Evaluation results are presented and discussed in §3.5, followed by the use of APPSHOT for a downstream service-level traffic dependent application in §3.7. Finally, §3.9 concludes the chapter.

3.2 Mobile Traffic and Context Data

For the purpose of modeling, analysis and evaluation in this work, we make use of a real-world mobile traffic dataset collected in the production network of a major mobile network operator in Europe. We also gather data for a variety of contextual attributes for the target regions from public sources.

Mobile Traffic Data. Our traffic dataset spans 10 major cities in a European country (referred henceforth as CITY 1-CITY 10)¹, where it covers the mobile demands of the whole subscriber base of the operator, amounting to around 30% of the local user population. This data was obtained by monitoring individual IP data flow sessions in the operator’s network over the General Packet Radio Service (GPRS) Tunneling Protocol User plane (GTP-U). To infer the services corresponding to the traffic flows, the operator employs a combination of proprietary and commercial traffic classification tools on top of Deep Packet Inspection (DPI) probes, which allows identifying a very wide range of mobile services with a high degree of accuracy [159]. Note that the data was aggregated geographically (per antenna sector) and temporally by the operator, so as to make the data non-personal and to preserve user privacy; all operations were carried out within the operator premises, under control of the local Data Privacy Officer (DPO), and in compliance with applicable regulations, according to GDPR (General Data Protection Regulation) regulations [3]. The data was aggregated over all users in space and time in secure servers at the operators’ premises, and we only accessed de-personalized aggregates.

Each city is represented in the data as a regular grid tessellation over space with each grid cell (*i.e.* pixel) covering $250 \times 250 m^2$. Unsurprisingly, different cities have different geographical sizes in terms of number of pixels in each dimension, and range from 33×33 to 97×123 pixels. Traffic data per pixel consists of overall mobile traffic volume *for each service* across uplink and downlink directions in bits/s, over time. The dataset covers a continuous period of 6 weeks. In this dataset, we consider the top 10 popular services that contribute to more than 80% of the total traffic volume, namely: YouTube (YT), Instagram (INS), SnapChat (SC), WhatsApp (WA), Netflix (NF), Apple Store (AS), iTunes, Facebook (FB), Twitter (TW), and Google Play (GP). As such, the effective total mobile traffic in our study is the sum of traffic due to these top-10 services.

Context Data². Our conditional generation model takes advantage of contextual

¹We can not disclose any information about the actual name of cities and the specific region in them. All data information is reserved as confidential by the data provider

²To align the contexts with traffic map on pixel level, we follow the method used in [159]

attributes to produce credible synthetic traffic. We gather a wide range of context data from easily accessible public sources, so that the method is applicable as widely as possible. All attributes for each city are mapped to the corresponding regular grid tessellation used to represent mobile traffic data, examples under each attribute are illustrated in Figure 3.1. In all, we consider 27 different contextual attributes, as outlined below.

Population. The number of inhabitants residing in each grid cell, as reported in the relevant national census.

Land Use. The different uses of the land within each grid cell, obtained from the Copernicus Urban Atlas repository [12]. We only retain land use types that exhibit non-negligible correlation with mobile traffic (as per Spearman’s correlation coefficient (SCC) [119]). Ultimately, 12 land use attributes are considered, listed in Table B.1.

Points of Interest (PoIs). The number of landmarks of a specific class within each grid cell, extracted from the OpenStreetMap (OSM) repository [131]. We use a similar correlation analysis with traffic as above, and retain the 14 significant PoI categories (Table B.1).

Contextual Attribute	Avg. SCC
Population	0.639
Continuous Urban	0.220
High Dense Urban	0.180
Medium Dense Urban	0.128
Low Dense Urban	0.254
Very-Low Dense Urban	0.102
Isolated Structures	0.051
Green Urban	0.325
Industrial/Commercial	0.252
Air/Sea Ports	0.321
Leisure Facilities	0.322
Barren Lands	0.067
Sea	0.072
Tourism	0.135
Cafe	0.002
Parking	0.2110
Restaurant	0.1797
Post/Police	0.118
Traffic Signal	0.430
Office	0.343
Public Transport	0.080
Shop	-0.018
Primary Roads	-0.074
Secondary Roads	-0.009
Motorways	0.254
Railway Stations	0.371
Tram Stops	0.158

Table 3.1: List of contextual attributes considered.

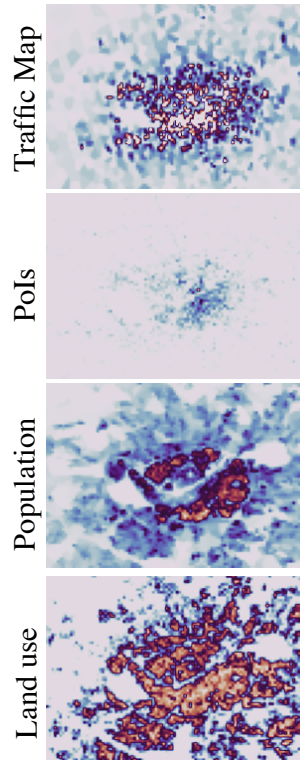


Figure 3.1: Spatial distribution of total traffic of all different services in CITY 1 and 3 selected context attributes.

3.3 Analysis of Mobile Traffic Characteristics Across Services and Cities

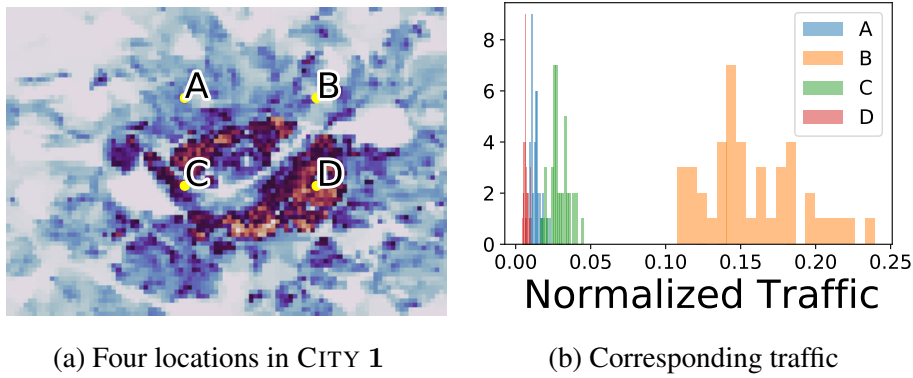


Figure 3.2: Illustration of inherent variation in traffic at each location over time, considering CITY 1 as an example.

In order to better inform the design of our generator, we first investigate the properties of mobile network traffic at the service level, across a number of different dimensions.

Relationship between Context and Traffic. We start by investigating how the traffic across the 10 target services relates to contextual information. A first important observation concerns the inherent stochasticity of mobile traffic: Figure 3.2b shows the distribution of total traffic observed over time at four different pixels in CITY 1, whose locations are shown in Figure 3.2a, and the traffic is normalized by the maximum pixel scale traffic (maximum value of traffic map over all dates) as displayed in the X-axis of Figure 3.2b. Note that mobile traffic can exhibit substantial variation at a location even though the corresponding context remains the same: this is, *e.g.*, the case of the population density illustrated in Figure 3.2a. In addition, the correlation between mobile traffic and contextual attributes for any given region is non-trivial. This is as exemplified in Figure 3.1, where three sample contextual attributes do not show any obvious visual correlation with the mobile traffic.

Takeaway message. The generation process must capture the stochastic nature of mobile traffic, by correctly modeling the relationship between static context information and spatial traffic demand at different time periods. Also, the lack of simple correlations between individual contexts and traffic indicates that a naive univariate statistical model based on any one attribute is not an effective generator, thus motivate the more complex multivariate designs we consider.

Correlations with Service Level Traffic. The above analysis considers aggregate traffic. As we are interested in service-level generation, we now examine the dependence of the demand for individual services on the various contextual attributes. Figure 3.3 shows correlation between the traffic snapshots of different services and the contextual attributes in three cities. We observe that, for a given city, the correlation between different services and any single attribute is close – each column generally has a similar color, but the service-context correlation varies across attributes (columns). This hints that the spatial distribution of traffic is consistent across services in a same

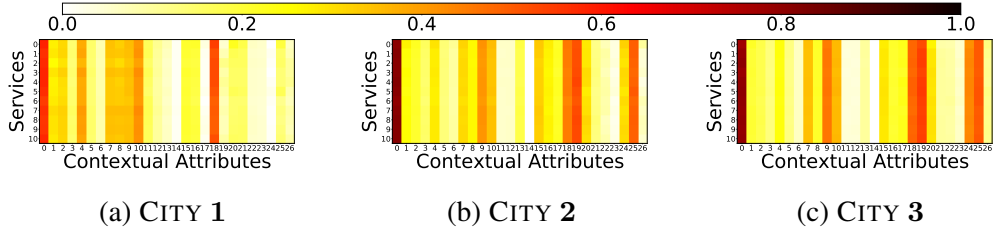


Figure 3.3: Correlation between the traffic of mobile services and contextual attributes in three different cities.

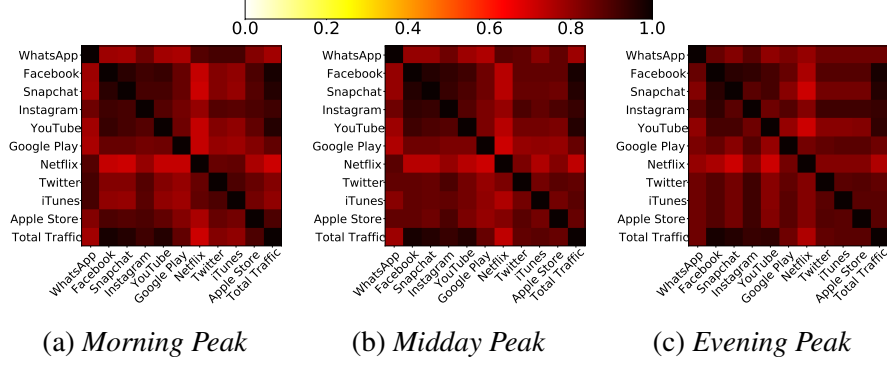


Figure 3.4: Pairwise similarity between traffic snapshots of different mobile services (as per SSIM) in CITY 3.

city. A more detailed analysis of cross-service traffic similarity further corroborates this observation: in Figure 3.4, we use structural similarity (SSIM) [175] measure to compute the spatial similarity between the spatial demand of pairs of services, for different daily peak hours in the morning, midday and evening. Note that SSIM is a classical image fidelity metric, which allows comparing individual pixels between a pair of images (here traffic maps of a pair of services) while also accounting for the differences in the whole spatial construct across the compared images. As shown in the plots, the spatial variations between different services stay relatively consistent at all times. Yet, not all service-level demand pairs display the same level of similarity, as SSIM between different service pairs ranges from 0.55 to 0.95 for any given time period.

Takeaway message. The diverse correlations among services indicate that naive transformations (*e.g.*, scaling) are insufficient to generate traffic snapshots for one service from the snapshots of a different service. However, more complex transformations may still take advantage of the significant but varying degree of similarity among the traffic of individual services. This suggests a model design that natively performs a joint synthesis of all per-service snapshots.

Traffic Characteristics in Different Cities. Figure 3.3 also suggests that the relationship between service-level traffic and contextual attributes is different across different cities. The heterogeneity among cities also appears in terms of average daily traffic volume, depicted separately for weekdays and weekends in Figure 3.5. Population, city size, and user preference, all contribute to such heterogeneity. For instance, CITY 1 has significantly higher traffic volume, about six to twenty times that of other cities. The traffic generation model must be able to capture such traffic heterogeneity across different regions. We also notice that traffic demand during weekdays is around

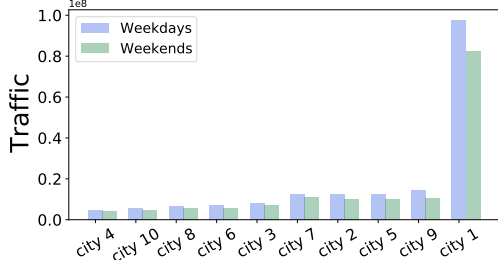


Figure 3.5: Average daily traffic.

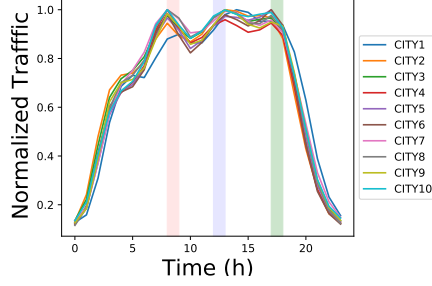


Figure 3.6: Total traffic.

20% higher than weekends for all cities. Our evaluations therefore highlight weekday traffic generation but relative performance results across different methods are similar for weekends.

In contrast, Figure 3.6 shows that such differences do not emerge at the level of aggregate normalized daily traffic, which is very consistent across all cities. Specifically, we identify the same three peak hours for all cities: in the morning (8-9am), around midday (12-1pm), and early in the evening (5-6pm).

Takeaway message. Generalizing the traffic generation task across cities is a significant challenge, as context-traffic correlations are highly diverse between cities. So the model must be designed so as to facilitate such generalization, which shall also be a key element of the performance assessment. Also, in our evaluation we will focus on the three peak hours identified above, as they are consistent across cities and especially important for, *e.g.*, network planning or network resource management purposes.

3.4 APPSHOT

Based on the insights from §3.3, the generation of high-quality multi-service traffic snapshots faces the following major challenges: 1) synthesizing high-fidelity traffic snapshot from context input with significant statistical variation; 2) preserving correlations among multiple services, both in terms of structural similarity and percentage contribution to total traffic; 3) allowing traffic generation for target cities of arbitrary spatial sizes; and 4) accommodating diverse traffic characteristics and context data ranges across cities.

With APPSHOT, we tackle challenges 1) and 2) by designing a tailored conditional deep generative model (§3.3), and solve challenges 3) and 4) via customized data processing and training methods (§3.4.2) and hyper-parameter tuning.

3.4.1 Problem Statement

Let $\chi = \{X^1, X^2, \dots, X^N\}$ be a real-world mobile network traffic dataset that contains sets of observations of mobile traffic, such that each set is collected in a different geographical region, *i.e.*, city. The data for each city $n \in \{1, \dots, N\}$, includes observations over a given span of time T^n , hence $X^n = \{x_1^n, \dots, x_{T^n}^n\}$. The observation at each time slot is composed of traffic due to S different services, *i.e.*, $x_t^n = \{x_{t,1}^n, \dots, x_{t,S}^n\}$. For time slot $t \in 1, \dots, T^n$ and service $s \in 1, \dots, S$, we rep-

represent the mobile traffic observation $x_{t,s}^n \in \mathbb{R}^{H^n \times W^n}$ as a single channel image, whose pixels map to the spatial units over which network traffic is recorded. Then, each pixel value corresponds to the network traffic value recorded for service s at a specific geographic location; and, $H^n \times W^n$ are the height and width of the city n 's dimensions in pixels, respectively; the dimensions may differ between cities.

In addition, each observation $x_{t,s}^n$ is associated with a set of K conditions, *i.e.*, publicly available contextual attributes that may explain the volume of traffic generated by mobile users (*e.g.*, as Figure 3.1 illustrates, population distribution in the region, land use characteristics, presence of points of interest, *etc.*). We denote the set of conditions for each city as its *context*, and represent it as the set $C^n = \{c_1^n, \dots, c_K^n\}$. For each condition $k \in \{1, \dots, K\}$ in city n , we have $c_k^n \in \mathbb{R}^{H^n \times W^n}$, and thus $C^n \in \mathbb{R}^{K \times H^n \times W^n}$, which is a multi-channel image with one channel per attribute. Note that the static, spatial contextual attributes alone are typically insufficient to fully explain the corresponding network traffic, as illustrated earlier in Figure 3.2b.

Our goal is to synthesize network traffic data $f_{t,s}^m \in \mathbb{R}^{H^m \times W^m}$ for an unseen region m at a particular time t and given context C^m in a way that the synthetic $f_{t,s}^m$ samples exhibit similar data characteristics to the real training data χ and are compatible with the provided C .

3.4.2 Patch based Learning Methods

In order to optimize the learning process, the mobile traffic data and contextual attributes need to be carefully formatted, as presented next.

Patching and Formatting

To create the training samples, we divide the $H^n \times W^n$ traffic map $x_{t,s}^n$ of each city n and service s in time slot t into smaller patches $x_{t,s}^{n,l}, l \in \{1, \dots, L\}$, where L is the total number of patches. This formatting has two advantages. Firstly, cities vary in their geographical span and so their traffic maps have varied dimensions, hindering the design of a single model that can handle different sized cities: here, employing fixed smaller sized traffic patches allows using the same generator model architecture regardless of the city dimensions considered for training or generation. Secondly, it allows using diverse traffic patches from different snapshots together to enable a more efficient training via stochastic gradient optimization. Moreover, different local sub-regions of a same city can have similar relationship between the context and traffic. So training at the patch level can be seen as a form of *weight-sharing* – a type of *regularization* technique – to let the model learn the actual casual relationship between context and traffic instead of memorizing the mapping.

The output synthetic traffic generated at the patch level (denoted as $f_{t,s}^{m,l}$) can be of a high quality for each individual patch but may leave artefacts at the boundary of patches when sewing the patch level outputs to a city-level traffic map. To overcome this issue, we associate with each traffic patch $x_{t,s}^{n,l}$ a *trimmed* context patch $c_k^{n,l}$ (identical across all attributes) that includes a margin around the traffic patch (see Figure 3.7a). This is considering that only a portion of the city-wide context that is in the geographical vicinity of the traffic patch stays relevant to the learning process. Crucially, the additional margin ensures that the border pixels of a traffic patch $x_{t,s}^{n,l}$ have

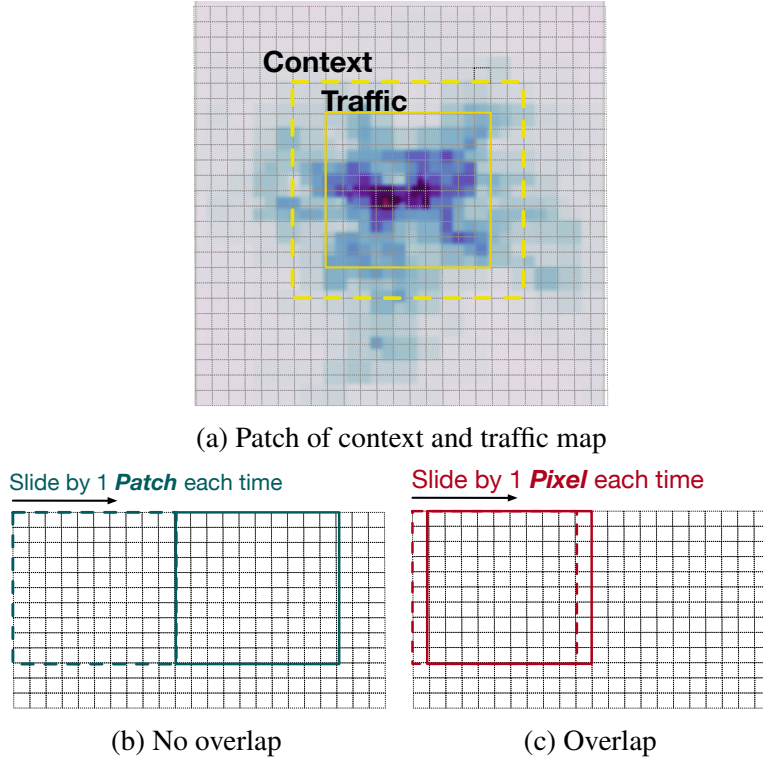


Figure 3.7: (a) Traffic patch and corresponding context patch; (b) non-overlapping and (c) overlapping traffic patch cases.

sufficient context during the learning process. Clearly, the number of context patches is the same as that of traffic patches, and we denote by $c_k^{n,l}$ the complete context patch corresponding to $x_{t,s}^{n,l}$.

As an additional measure towards artefact-free synthetic traffic maps, we consider overlapping traffic patches as shown in Figure 3.7c and slide across them by 1 pixel each time during training and generation. Compared to the straightforward ‘no overlap’ case illustrated in Figure 3.7b, each pixel in the output traffic map benefits from being part of multiple traffic patches. This not only helps with avoiding edge effects but also serves as a data augmentation method. For example, with a city map of 20×20 and traffic patch size of 10×10 with dimensions in pixels for both, no overlap case yields just 4 patches. Overlapping case, on the other hand, results in 121 patches³ for the same example. More effective data for training aids in capturing key spatial traffic features at high fidelity but also helps with better generalization across diverse cities.

Normalization

Given that different services may have vastly different traffic volumes, the training and generation for services with relatively lower volume can become an issue if not handled properly, especially in a multi-channel CNN model where the weights involved in the generation of distinct services are broadly shared. To guard against this issue, we employ service-level traffic normalization as a pre-processing step. Specifically, we

³We get 121 patches for this example by considering traffic patches with their top-left most pixel falling at each of the pixels in the $[1, 1]$ to $[11, 11]$ square region.

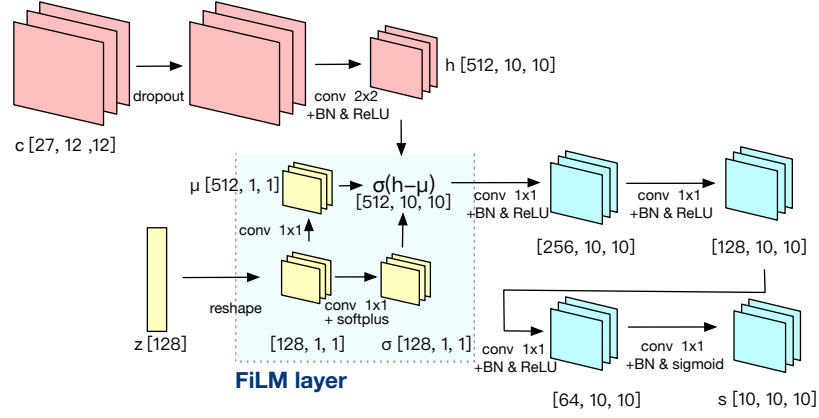


Figure 3.8: Schematic of APPSHOT generator architecture, labeled with input and output at each layer.

normalize the traffic values, x_s , of each service s by dividing them with $x_{s,\max}$, *i.e.*, the *global* per-pixel maximum traffic volume value observed for s .

This results in traffic values for each service to independently fall between 0 and 1. As part of this normalization step, we also add a small ϵ value to handle cases where no traffic is recorded for a service. The above normalization step can be easily reversed during post-processing on the output synthetic traffic map via a rescaling step.

3.4.3 Detailed Model Design

Generator

The generator in APPSHOT is responsible for generating traffic map of all services, each represented as a different channel in the multi-channel image output.

The generator, denoted as G_θ with θ representing the weights of the neural network, is a conditional latent variable model instantiated by a CNN based architecture. Formally, we use the latent variable z to model stochasticity and unobserved conditions, then the probability of observing an actual mobile traffic x given conditions c is modeled as $p_\theta(x|c) = \int p_\theta(x|c, z)p(z)dz$, where θ represents the parameters of the conditional probability. Figure 3.8 shows a schematic of the generator’s neural network architecture in APPSHOT. An important remark is that context c is spatial, whereas z is non-spatial. The non-spatial input z in APPSHOT is processed via a specialized FiLM conditioning layer [136], which effectively creates two convolutional entry branches in the initial stages of the generator network. This design avoids the risks of a naive conditioning on the latent variable z (*e.g.*, simply concatenating it with c) that can lead to the network completely ignoring stochasticity. Instead, the FiLM ensures that the latent variable is duly accounted for in the generation process. The result of the separate convolutional branches are then merged via an affine transformation into a hidden representation whose spatial dimension is same as the output traffic map. This representation is then processed by stacked convolution layers with size-1 kernels to produce the final sample s .

In Figure 3.8, we label the dimensions of input/output at each layer using the format of $[channels, size_x, size_y]$. For example, the $c[27, 12, 12]$ on the input side means the context input is a multi-channel image with 12×12 dimensions and 27 channels; each channel here represents a particular condition (*i.e.*, contextual attribute). To re-

duce over-fitting to the conditions, we add a channel-wise dropout layer to the input conditions (with a dropout rate of 0.02). The first convolutional layer has a kernel size of $N_c - N_x + 1$. Here N_x refers to the dimension of the output traffic patch size (empirically set to 10) whereas N_c is the context patch dimension (empirically set to 12 to provide the best average output quality $N_x = 10$). The rest of convolutions are of size 1. As per the number of channels: $N_c \rightarrow 8F \rightarrow 4F \rightarrow 2F \rightarrow F \rightarrow 1$, where the base number of features F are 64. For the FiLM layer process, the latent variable z is a N_z dimensional noise vector with N_z set to $16F$. All intermediate activations are ReLU [120], following a batch normalization (BN) layer [78]; the final activation is Sigmoid. For σ from the FiLM layer, we use the softplus activation, $F(x) = \log(1 + \exp(x))$, to ensure it is positive.

Training

To learn θ , we train the model by optimizing the loss function, as elaborated below. The generator is trained in an adversarial manner with two discriminators to reflect correlation among services and their contribution to total traffic. For this purpose, we define a conditional probability distribution $p_{\mathcal{D}}$ based on real data (ground truth traffic) and corresponding context for cities 1 to N in the training data (*i.e.*, $\{(x^1, C^1), \dots, (x^N, C^N)\}$). We then find the model weights θ^* that minimize a divergence criterion between the data distribution $p_{\mathcal{D}}$ and the model p_{θ} . Specifically, following standard GAN formulations [60], we train the model by minimizing the Jensen-Shannon (JS) divergence, *i.e.*, $\theta^* = \operatorname{argmin}_{\theta} JS[p_{\mathcal{D}} || p_{\theta}]$. One of the discriminators called **individual quality discriminator** (D_1) is designed to evaluate the overall fidelity of the multi-service traffic map. For this discriminator, the adversarial loss is defined as:

$$\mathcal{L}_{JS}^{D_1}(p_{\mathcal{D}, p_{\theta}}) = \mathbf{E}_{p_{\mathcal{D}}}[\log D_1(x, c)] + \mathbf{E}_{p_{\theta}}[\log(1 - D_1(\tilde{x}, c))].$$

where the $\tilde{x}$ is the synthetic traffic map of x .

Unlike conventional works on image generation that treat different channels independently, we need to capture the correlation between different channels. We also need to minimize the divergence between the sum of output channels and real total traffic at the pixel level. By training each channel to target generation of synthetic traffic maps for a different service does not ensure the correct sum of all traffic maps from different channels, so providing extra regularization is helpful in our case. To constrain the sum of traffic from different services and encourage the model to learn the correct composition of total traffic, we introduce a second discriminator called **sum quality discriminator** (D_2) with its adversarial loss defined as:

$$\mathcal{L}_{JS}^{D_2}(p_{\mathcal{D}, p_{\theta}}) = \mathbf{E}_{p_{\mathcal{D}}}[\log D_2(\sum_{s=1}^S x_s, c)] + \mathbf{E}_{p_{\theta}}[\log(1 - D_2(\sum_{s=1}^S \tilde{x}_s, c))],$$

where S is the total number of services under consideration, x_s refers to the traffic map of service s within S . $\tilde{x}_s$ means the synthetic traffic corresponds to ground truth x_s .

Training solely with adversarial training as described above is insufficient, which generally leads to higher training instability and lower fidelity output. So in APPSHOT, besides adversarial training, we make the training process more stable and controllable by adding the L1 loss. Specifically, we use the **L1 norm** of the synthetic multi-channel

traffic map (with respect to its real counterpart) as part of the loss function. L1 loss function is shown to be empirically effective in prior work (e.g., [147]).

As the overall loss function of the generator, we take the sum of the above two adversarial losses and the weighted supervised learning loss (L1 norm):

$$\mathcal{L} = \mathcal{L}_{JS}^{D_1}(p_{\mathcal{D}}, p_{\theta}) + \mathcal{L}_{JS}^{D_2}(p_{\mathcal{D}}, p_{\theta}) + \lambda \mathbf{E}_c \{ ||\mathbf{E}_{x \sim p_{\mathcal{D}}}[x], \mathbf{E}_{x \sim p_{\theta}}[\tilde{x}]||_1 \}.$$

This final loss $\mathcal{L}$ is used to update the discriminators and generator in turn. Here λ is a tuneable parameter to adjust the weight of L1 loss; we set $\lambda = 0.5$ by default in our tests.

3.5 Performance Evaluation

3.5.1 Fidelity Metrics

Weighted Error (WErr). This metric quantifies the composition of a synthesized multi-service mobile traffic dataset relative to the corresponding real (ground-truth) data. Suppose in a real dataset made up of traffic from multiple services, the actual percentage of traffic due to a service s among S services with respect to total traffic is r_s and its traffic volume is t_s . If the traffic volume of the same service in the corresponding synthetically generated dataset is $\tilde{t}_s$, then the Weighted Error (**WErr**) is defined as:

$$\text{WErr} = \sum_{s=1}^S r_s \frac{|t_s - \tilde{t}_s|}{t_s}.$$

In other words, it is the relative estimation error in traffic volume per service weighted by each service’s actual percentage, averaged over all services. Smaller WErr means more accurate service composition in the synthetic dataset.

Normalized EMD (NEMD). Earth Mover’s Distance (EMD), also known as Wasserstein Distance [148], is a distance function defined between two probability distributions over a given metric space (e.g., 1D, 2D). It has been used in similar settings as ours, e.g., to assess the quality of GAN models [64], or to compare two spatial distributions [79].

For our particular purpose of comparing real and synthetic service-level traffic maps, EMD is sufficient when we focus on a particular city. But that is not true for comparison over a set of cities due to their widely different sizes. To address this issue, we normalize the EMD between real and synthetic maps by the EMD between real map and uniform (2D) distribution. Let us denote the uniform traffic map as ϕ , the real map in simplex space as μ , and the synthetic map in simplex as v ; then, the normalized EMD (NEMD) is defined as:

$$\text{NEMD} = \frac{\text{EMD}(\mu, v)}{\text{EMD}(\mu, \phi)},$$

where $\text{EMD}(a, b)$ is the EMD between 2D distributions a and b . It is worth noting that with EMD, the images will be converted to simplex space, and thus the information of the original data range is lost. This calls for use of complementary metrics such as

those outlined next.

SSIM and PSNR. We also consider Structural Similarity Index Metric (SSIM) and Peak Signal-To-Noise Ratio (PSNR) – the two commonly used image quality assessment metrics [68] – to respectively evaluate the structural and pixel-level fidelity.

Hotspot Histogram EMD (HEMD). As noted earlier, hotspots are a key spatial feature of interest with mobile traffic data. To quantify the extent to which different methods faithfully capture this feature, we use the EMD between 1D distribution (histogram) of hotspots in synthetic and real data.

Besides the above quantitative fidelity metrics, we also consider qualitative measures including traffic histograms at city and pixel level as well as for number of hotspots to visualize the quality of the synthesized data with different methods, especially to gauge their ability to model underlying data variations (stochasticity).

3.5.2 Baselines

We consider a wide range of baseline methods to comparatively evaluate APPSHOT in terms of the metrics above. The primary selection of baselines includes many image based traffic generator. The APPSHOT project primarily processes input in the form of images, a decision rooted in its foundational design constraints. This approach is not arbitrary; rather, it is supported by evidence from Reference [159], where the authors demonstrate the efficacy of using image-based inputs to integrate diverse contexts effectively. Building upon the methodologies outlined in Reference [159], an optimal strategy for this project would be the implementation of an image transformer, such as Pix2Pix [80], which is well-suited for handling image-centric data. In contrast to APPSHOT, when considering other model-based traffic generators like FDaS [43], it becomes evident that omitting explicit modeling of spatial correlations leads to sub-optimal results. These models, while adept in certain aspects, fall short in capturing the intricacies of spatial relationships inherent in network data. This limitation underscores the significance of incorporating spatial correlation in modeling to achieve more accurate and satisfactory outcomes in network traffic generation.

In our evaluation of baseline models, we also took into account non-generative approaches. For example, the Pix2Pix model, when operating under highly detailed conditions, tends to produce outputs with limited variation, which effectively positions it as a non-generative model in our analysis. Additionally, we explored simpler methodologies, such as plain Convolutional Neural Networks (CNNs). The collective findings from these experiments clearly indicate that generative models play a crucial role in effectively addressing the challenges of traffic map generation. The use of generative models has shown to be instrumental in capturing the complexities and variabilities inherent in this task, something that more straightforward or non-generative approaches have struggled to achieve.

CNN based Regression. A simple-minded approach for our multi-service traffic map generation task is to train a deep neural network (DNN) that takes conditions c as input and predicts a multi-channel image output x . Given the spatial nature of the output, CNN based regression is a good choice. This approach clearly fails to model stochasticity, a key characteristic of mobile traffic data. We implement this baseline via CNN on U-net architecture [147], and perform patch learning with non-overlapping fixed size patches with patch dimensions same as in APPSHOT.

	APPSHOT			Data	CartaGenie	SpectraGAN	Pix2Pix	CNN	FDaS
	Morning	Midday	Evening	All Peaks	Morning	Morning	Morning	Morning	Morning
WErr ↓	17.93%	19.09%	18.98%	15.10%	24.85%	17.74%	38.81%	42.07%	63.67%
NEMD ↓	0.35	0.36	0.35	0.23	0.44	0.57	0.52	0.58	0.99
SSIM ↑	0.84	0.86	0.86	0.96	0.85	0.79	0.78	0.76	0.44
PSNR ↑	34.88	32.61	32.47	39.24	36.15	35.01	34.07	34.88	29.33
HEMD ↓	2.09	1.99	2.24	0.6	3.90	2.93	4.31	5.79	9.20

Table 3.2: Fidelity performance of APPSHOT at different peak periods (left) and of baselines for morning peak period (right).

Pix2Pix [80]. This model has been successfully used for image transformation tasks in computer vision. It is both conditional and stochastic like conditional GANs, but makes use of a tailored DNN architecture for image-to-image translation. Pix2Pix has several limitations compared to our APPSHOT approach. In our implementation of this baseline, we perform non-overlapping patch based learning as above.

Fit Distribution and Sample (FDaS) [43]. A prior approach for mobile traffic data generation essentially involves fitting an empirical distribution to model the traffic data using maximum likelihood estimation of parameters and then sampling it afterwards to generate synthetic traffic [43]. While only total traffic demand across all services is considered in [43], we apply their approach separately for each service to allow comparison. Like in [43], we find log-normal distribution best fits the data but with different parameters across distributions, as expected. This approach has the inherent limitation of not being able to capture traffic correlations in space or time.

CartaGenie [178] and **SpectraGAN** [180]. These are the state of the art mobile traffic data generation methods that also employ conditional deep generative modeling as in APPSHOT. They, however, target generation of spatial snapshot or spatiotemporal data for *total* traffic, as with the FDaS baseline above. To apply them to the multi-service traffic generation case studied in this paper and have them as baselines, we train and use multiple separate instances of CartaGenie and SpectraGAN models, one per each service.

Data. In addition to the above baselines, we also consider an ideal case for reference, which we refer to as “Data”. Metrics for this case are computed by splitting the real dataset (with 30 weekdays) into two distinct subsets (15 weekdays each part), and comparing these subsets of real data against each other. This captures the variability of the dataset within itself, which is a proxy for the ‘upper bound’ fidelity performance a synthetic data generation model like APPSHOT can achieve.

3.6 Results

In the performance evaluation, we try to fine tune the baselines as well, including changing the number of parameters or layers, select the hyper parameters with the best overall performance. The configuration of all the baselines fine tuned to this task to the best of our knowledge.

3.6.1 Fidelity and Generalization

Throughout this section, we consider a leave-one-city-out evaluation. Specifically, each of the 10 cities in our dataset is taken as a test city in turn while using the data for the remaining 9 cities as the training set. This type of evaluation lets us assess the ability of APPSHOT and various baselines to generalize to unseen cities as well as their ability to handle different sized cities and their differences in traffic/context data value ranges. Following the earlier analysis in §3.3, our evaluations focus on weekdays and morning/midday/evening peak hours. For brevity, we mainly show results for the morning peak hour period, unless otherwise specified; but similar conclusions apply for other periods.

Correlation between services. As shown in §3.3, traffic for different services exhibit strong mutual correlations. So it is important for the generated traffic data to preserve this feature. To assess APPSHOT on this aspect, for each test city in the dataset, we compute the average of SSIM between traffic snapshots of every pair of services in the real ground-truth data, and similarly in the data synthesized with APPSHOT. We then compute the absolute error in the average pairwise SSIM computed over synthetic data with respect to that on real data. Results shown in in Figure 3.9a indicate that APPSHOT yields a small error, within 14% of the real data on average.

Composition of different services for different cities. Besides maintaining inherent correlations between traffic for different services, it is also important to ensure that their proportions relative to total traffic are preserved in the synthesized data. Figure 3.9b shows the error on this measure with APPSHOT relative to real data for different services with each test city. We observe that APPSHOT yields a low error within 20% of real in most cases. WhatsApp case is the only exception but this is an artefact due to traffic for this service making up a very small percentage (0.5%) so small absolute errors appear as big relative errors.

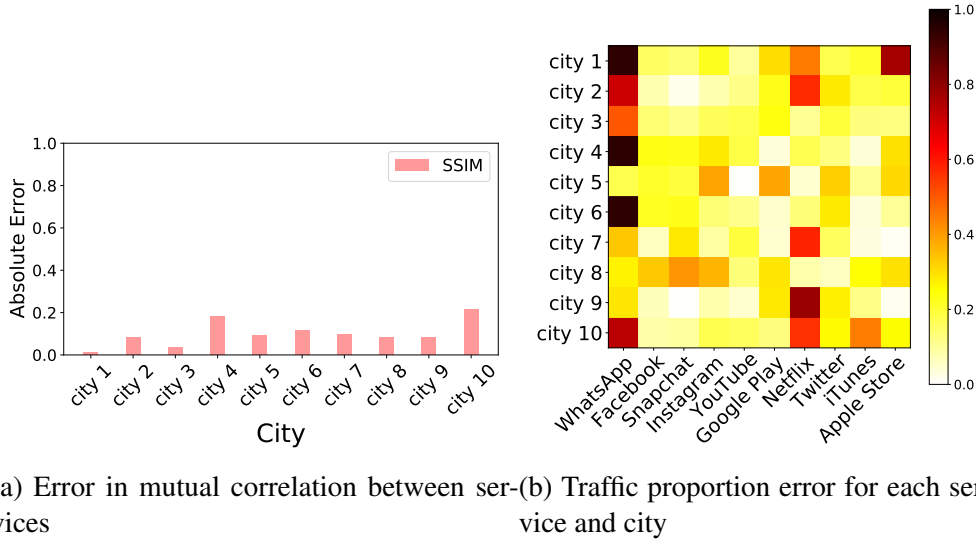


Figure 3.9: APPSHOT service-level performance across cities.

Performance relative to baselines. The results of the comparative evaluation for the morning peak hour period are summarized in Table 3.2. We observe that APPSHOT, in the comparison with the baseline methods (CartaGenie, SpectraGAN, Pix2Pix, CNN

and FDaS), yields the best performance on two of the metrics (NEMD and HEMD) while being close to the best result on the other three metrics. Overall, APPSHOT provides the performance closest to the ideal ‘Data’ reference across all metrics. Among the baselines, FDaS is clearly the worst performer on all metrics, showing the limitations of this approach in handling correlations in traffic and ensuring fidelity of the service-level snapshots.

Two other baselines – Pix2Pix and CNN based regression – have somewhat similar performance on all metrics, but considerably worse on most metrics relative to APPSHOT. This highlights their inability to accurately capture the spatial distribution of traffic relative to ground truth, which particularly harms the way certain key characteristics in the generated data (*e.g.*, the position and number of hotspots). This is particularly reflected in the HEMD performance which is more than double (twice as worse) than APPSHOT. Since Pix2Pix is marginally better than CNN based regression on all metrics, we only consider the former in the rest of our evaluations.

SpectraGAN exhibits slightly better performance than APPSHOT with respect to two metrics (WE_{err} and PSNR) but substantially worse on the remaining three metrics. CartaGenie is similar in that it does slightly better than APPSHOT with respect to SSIM and PSNR but has substantially worse performance on the other three metrics. Broadly speaking, this overall relatively poor performance of SpectraGAN and CartaGenie compared to APPSHOT can be attributed to their inability to exploit inter-service correlations due to independent generation of per-service traffic and insufficient measures to correctly model hotspots (reflected in their significantly worse performance in terms of HEMD).

The shortcomings of these two baselines with respect to APPSHOT are apparent in the visualizations of synthetic traffic maps they generate as shown in Figure 3.10. We see that CartaGenie and SpectraGAN respectively yield unacceptable synthetic traffic maps for Instagram and YouTube for CITY 1, the most challenging target city given its vastly bigger size, population density, traffic volume and hotspots compared to other cities in our dataset (see §3.3). In §3.6.2, we will further explore the performance of CartaGenie and SpectraGAN relative to APPSHOT. In Figure 3.10, also note that Pix2Pix fails to provide meaningful traffic maps for any service and exhibits severe artefacts, consistent with its poor performance in terms of quantitative fidelity metrics as seen above.

Performance at different peak periods. We now consider how well APPSHOT generates service level traffic snapshots at different peak periods. Results for different fidelity metrics averaged across all test cities are summarized in the left panel of Table 3.2. We observe that APPSHOT provides consistent performance for all periods close to the ideal ‘Data’ reference, with WE_{err} under 20% and near-ideal results for SSIM and PSNR.

Capturing statistical variations. It is important for a synthetic mobile traffic data generation model to model inherent stochasticity in such data. This reflects the model’s ability to learn traffic distributions conditioned on the contextual input, rather than simply outputting a deterministic transformation (as CNN based regression would do). We examine this aspect considering histograms of city-level and pixel-level total traffic volume across all test cities, and the number of hotspots. Results shown in Figure 3.11 for APPSHOT clearly demonstrate that it achieves this intended goal. Note that we include pixel-level histograms for only two arbitrarily selected test cities for brevity.

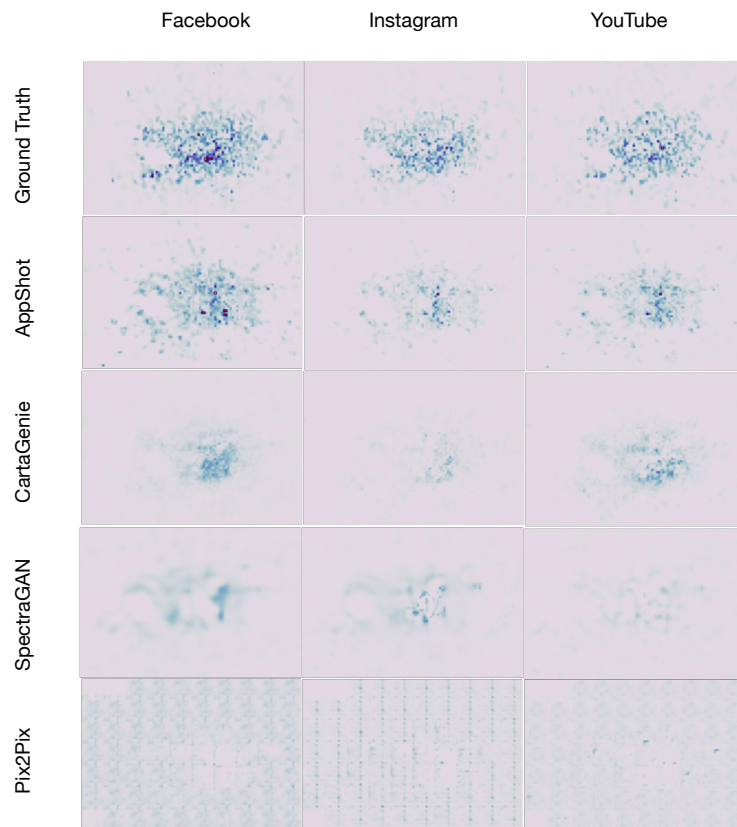


Figure 3.10: Synthetic traffic maps for select services in CITY 1 generated with different methods compared against the ground truth traffic maps corresponding to those services.

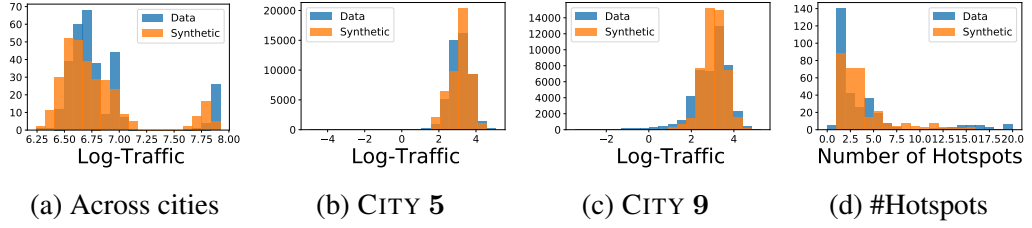


Figure 3.11: Statistical features of APPSHOT-generated data.

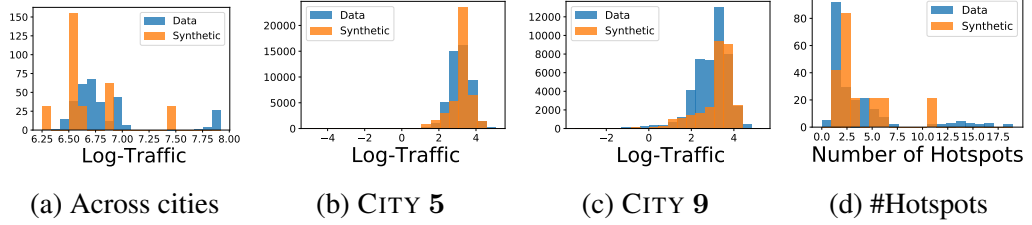


Figure 3.12: Statistical features of APPSHOT-generated data *without* the FiLM layer.

3.6.2 Detailed Comparisons with CartaGenie and SpectraGAN

CartaGenie

Earlier in this section, we have already highlighted the benefit of APPSHOT as a whole relative to the alternative of using multiple separate per-service instantiations of the CartaGenie model. Here we dissect APPSHOT to examine the benefit due to some of its underlying design choices and contrast with those underlying CartaGenie.

In Table 3.3, the ‘L1+D1+D2’ represents the APPSHOT design, using adversarial training with two discriminators as well as use of overlapping patches and sliding across them one pixel at a time (see Figure 3.7c). The ‘No Overlap’ case is different from ‘L1+D1+D2’ in that the former uses non-overlapping patch based training (see Figure 3.7b) as in the CartaGenie design. Clearly, non-overlapping patches worsens performance on all metrics, significantly so for several of the metrics (WErr, NEMD and HEMD).

The other two alternative designs – ‘L1 Only’ and ‘L1+D1’ – shown in Table 3.3 use overlapping patches as in APPSHOT but differ in their loss functions. Here ‘L1 Only’ represents the case where only L1 loss is used for the loss function as done in CartaGenie. We see that doing so results in overall worse performance compared to APPSHOT (*i.e.*, ‘L1+D1+D2’). In particular, using L1 loss alone is clearly insufficient to accurately model traffic composition (as measured by WErr) and capturing hotspot distribution (HEMD). Addition of a discriminator (via adversarial training as in GAN), shown as L1+D1, helps on both fronts. Yet another discriminator (L1+D1+D2) to ensure correct traffic composition, as we do in APPSHOT, provides the best performance overall.

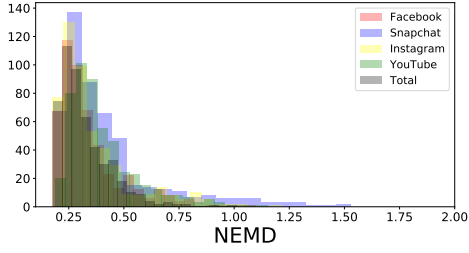


Figure 3.13: Per-hour NEMD histogram for APPSHOT.

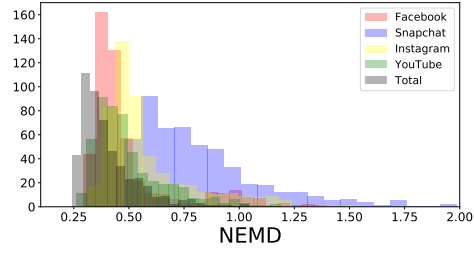


Figure 3.14: Per-hour NEMD histogram for SpectraGAN.

	No Overlap	L1 Only	L1+D1	L1+D1+D2
WErr ↓	32.04%	34.68%	22.68%	17.93%
NEMD ↓	0.47	0.33	0.36	0.35
SSIM ↑	0.79	0.81	0.85	0.84
PSNR ↑	32.6	37.3	35.2	34.88
HEMD ↓	3.70	5.59	2.19	2.09

Table 3.3: APPSHOT design (shown under L1+D1+D2 in the table) compared against alternative design choices. The ‘No Overlap’ and ‘L1 Only’ represent the design choices underlying the CartaGenie model.

SpectraGAN

Different from other baselines, SpectraGAN is designed to capture the spatiotemporal features of mobile traffic. We extend SpectraGAN to service-level generation by training it on each service independently. To evaluate APPSHOT in the time domain and show that it generalizes to different periods, we train APPSHOT to generate service level snapshots for each hour of the day (*i.e.*, the same granularity as SpectraGAN) separately, and obtain synthetic service-level traffic over time by stitching the hourly snapshots. Specifically, we train 24 models with APPSHOT that correspond to different hours of a day. The time series of city-scale total traffic of YouTube after stitching is illustrated in Figure 3.15.

Spatial-domain performance. The histogram of per-hour NEMD over a period of 3 weeks is shown in Figure 3.13 and Figure 3.14 for APPSHOT and SpectraGAN, respectively; there we consider total traffic and the four popular services. APPSHOT yields consistently good performance for individual services as well as for total traffic. SpectraGAN, on the other hand, performs significantly worse for some services, and is unstable over time. These results are in line with worse spatial fidelity (in terms of NEMD and SSIM) seen previously with SpectraGAN in Table 3.2. A key reason for this is its inability to exploit inter-service correlations, unlike APPSHOT.

Time domain performance. We employ the L1 distance of autocorrelation between synthetic and real data ($AC-L_1$), also considered in previous work [98, 180], to comparatively evaluate the temporal fidelity of the synthetic data between SpectraGAN and APPSHOT. Specifically, we compute this metric by taking the average value of L_1 norm between the corresponding points of the auto-correlations of real and synthetic time-series data, at the pixel level. Lower values thus imply better performance.

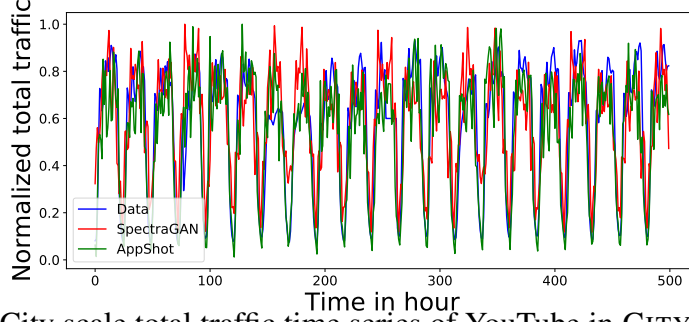


Figure 3.15: City scale total traffic time series of YouTube in CITY 1, normalized by the maximum hourly traffic.

	Instagram	Snapchat	Facebook	YouTube	Total (10)
APPSHOT	47.3	60.1	69.8	46.8	62.0
SpectraGAN	75.6	94.6	75.9	71.2	65.3

Table 3.4: Time domain performance comparison between APPSHOT and SpectraGAN in terms of $AC-L_1$ (lower is better).

Table 3.4 shows results comparing the performance APPSHOT with SpectraGAN in terms of median $AC-L_1$, considering the top four popular services and total traffic of all ten services available. While the two methods achieve similar performance for total traffic, APPSHOT has substantially better performance at the individual service level. Figure 3.15 highlights the particular case of YouTube traffic by way of explaining these performance differences. In Figure 3.15, we observe that SpectraGAN tends to largely overestimate the actual traffic relative to ground truth, especially during idle periods (e.g., in hours 180 or 335), while APPSHOT correctly models such situations.

3.6.3 Benefit from Other Design Choices and Parameter Tuning

We now present further results supporting design choices in APPSHOT, and discuss the tuning of its key hyper-parameters.

Noise Input Effect with FiLM Layer. As discussed in §3.4.2, naive conditioning on the noise input to the generator by simply concatenating it with (spatial) context input can cause the model to ignore the noise input altogether and prevent it from modeling stochasticity in the data. We avoid this issue by using FiLM layer [136] to provide the noise input separately through it. The benefit from this choice is highlighted by comparing the results in Figure 3.11 with that of Figure 3.12 where in the latter case APPSHOT uses naive conditioning on noise input without the FiLM layer.

Hyper-parameters. We have determined the best settings for various hyper-parameters empirically. These resulted in the use of 12×12 as input context patch size (for 10×10 output traffic patch size) and 3×3 kernel size at the first convolutional layer. Among the various hyper-parameters of the APPSHOT neural network model, through experiments and analysis, we find that the kernel size of initial convolutional layers in APPSHOT plays a critical role in determining the fidelity of output synthetic traffic maps (as illustrated in Table 3.5 and Figure 3.16). We choose 3×3 kernel size as the setting for the first convolutional layer that generally works well.

Kernel Size	CITY 1	CITY 2	CITY 3	CITY 4	CITY 5	CITY 6	CITY 7	CITY 8	CITY 9	CITY 10	Average
1	0.32	0.52	0.41	0.34	0.46	0.45	0.40	0.44	0.36	0.28	0.38
2	0.31	0.47	0.42	0.37	0.45	0.50	0.34	0.38	0.35	0.33	0.39
3	0.27	0.40	0.40	0.35	0.38	0.41	0.35	0.38	0.36	0.26	0.35
6	0.30	0.53	0.55	0.43	0.49	0.47	0.55	0.53	0.35	0.26	0.44
11	0.45	0.54	0.44	0.37	0.51	0.46	0.54	0.44	0.35	0.25	0.44

Table 3.5: APPSHOT performance in terms of NEMD with different kernel sizes for the first convolutional layer.

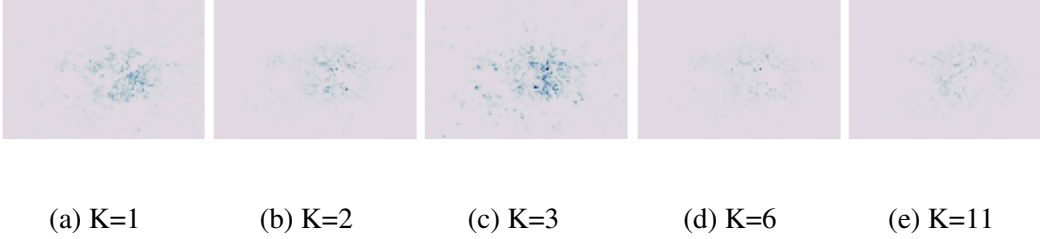


Figure 3.16: APPSHOT-generated traffic maps for Instagram in CITY 1 with different kernel sizes (K) for the first convolutional layer.

3.7 Use Cases of APPSHOT

As part of our performance evaluation, we assess the utility of APPSHOT through a downstream application use case of multi-service mobile traffic data. Specifically, we employ synthetic data generated by APPSHOT to effectively feed a recent model for the estimation of the multiplexing efficiency with radio network slicing [110]. The actual size of each pixel in our dataset is $250 \times 250m^2$, which is close to the coverage range of a small cell. We thus assume that the radio network is composed of small cells each matching one pixel. We further assume that each service is associated with an individual slice, *i.e.*, a dedicated and customized set of network resources and functions that allows achieving strong quality of service (QoS) guarantees to the service providers. The need to isolate resources to each slice (*i.e.*, service) is at the root of a reduced multiplexing efficiency: resources need to be allocated for each slice, and cannot be multiplexed as in legacy networks that cannot provide strong QoS [110]. In our case, the sliced resources are at the radio access level (*e.g.*, spectrum or baseband processing resources), hence must accommodate the per-service traffic generated in each pixel separately.

Formally, suppose there are N cells in the target region, and let us denote by $r_{i,s}(p, t)$ the minimal resource to serve the traffic demand of slice (*i.e.*, service) s in cell i for a fraction of time p over a reconfiguration period t . The value of $r_{i,s}(p, t)$ can be derived from multi-service mobile traffic data generated with APPSHOT and using the model in [110]. The (minimum) amount of resources needed to serve the overall traffic in absence of slicing (*i.e.*, when multiplexing across services is possible) is $R_i(p, t)$. Then the network slicing efficiency is:

$$E(p) = \frac{\sum_{t=1}^T \sum_{i=1}^N R_i(p, t)}{\sum_{t=1}^T \sum_{i=1}^N \sum_{s=1}^S r_{i,s}(p, t)}.$$

We compute the accuracy of estimating multiplexing efficiency with APPSHOT-generated data relative to using real data. We consider low and high coverage cases, respectively corresponding to covering 95% and 99% of demand in each reconfiguration period, *i.e.*, $p = \{0.95, 0.99\}$. In other words, more than 95% or 99% of the demand must be accommodated in each slice during a reconfiguration period. We consider a wide range of reconfiguration periods from 2h to 36h. As seen from the results in Table 3.6, the APPSHOT-generated data only introduces about 5% error in estimating the multiplexing efficiency compared with the real traffic data for short reconfiguration periods. The estimation error with APPSHOT data slightly increases with increasing reconfiguration period as well as lowered coverage probability but it always is within 10% relative to using real data.

	2h	4h	8h	12h	16h	20h	24h	36h
95% Coverage	4.7%	4.6%	5.6%	4.9%	5.5%	5.5%	8.1%	8.4%
99% Coverage	4.7%	4.6%	5.5%	4.9%	5.4%	5.5%	6.0%	6.8%

Table 3.6: Error in multiplexing efficiency estimation with APPSHOT-generated data for different coverage probabilities and reconfiguration periods.

3.8 Discussion

3.8.1 Limitations of APPSHOT

We recognize the potential impact of city-scale influences in certain scenarios, as evidenced in contexts similar to those observed in London, where unique urban effects are discernible. However, our batch generation approach presents challenges in fully exploring these city-specific correlations. The constraints stem from the limited availability of training data and the necessity to generate traffic maps across various scales, including arbitrary ones. Given these considerations, opting for batch generation emerges as the most pragmatic solution in our context. This methodology, despite its limitations in capturing localized urban dynamics, offers a balanced approach to handling diverse data availability and scalability requirements in traffic map generation.

Another notable limitation of our study is the inability to investigate the effects of varying grid sizes in traffic map generation, although we highlight the grid size is critical with few experiments in Figure 3.16 and Table 3.5. The diversity in traffic patterns suggests that adopting a multi-resolution approach, similar to techniques used in computer vision modeling, could potentially yield more nuanced insights. However, our exploration in this direction is hindered by a fundamental constraint: the lack of access to a diverse dataset with sufficient variations in density. Our training dataset is confined to specific countries and regions, where the density of variations is relatively uniform and we can therefore find an optimal grid size for all cities. This uniformity in our data limits our capacity to examine and understand the impact of different grid sizes on traffic map generation. The absence of such an analysis represents a significant limitation of our work, as it restricts our understanding of how varying densities might influence traffic patterns and predictions.

3.8.2 Potential Extensions

The objective of APPShot is distinct from tasks aimed at city-level completion. However, adapting APPShot for city-scale applications is a feasible and straightforward process. This can be achieved by dividing a city into several sections and training the model separately on these subdivisions. This method of segmenting a city and applying the APPShot model to each segment individually does not necessitate any alterations to the model itself. Essentially, this approach represents a direct and uncomplicated extension of APPShot’s capabilities, allowing it to operate effectively at a broader, city-wide level without requiring fundamental changes to its underlying methodology.

3.9 Summary

In this chapter, we have presented APPSHOT, a novel conditional deep generative model for synthesizing high-fidelity multi-service network traffic data that needs only publicly available context information of target regions. We have used real-world service-level mobile traffic data for multiple cities for our evaluation and show that APPSHOT not only outperforms a range of baseline approaches in terms of fidelity and also generalizes well to unseen regions. Our patch-based learning approach and the corresponding operations have proven to be effective in generating traffic for cities of different sizes. Also, data augmentation with overlapping patches significantly enhances performance with respect to handling traffic hotspots and diverse traffic ranges. The architecture of the APPSHOT neural network and the service-level constraints it incorporates significantly enhance the accuracy of service compositions in synthetic traffic, while preserving a strong structural correlation between services. Furthermore, APPSHOT is shown to capture realistic statistical variations on both city-wide traffic demand and structural characteristics (e.g., number of hotspots). Finally, we have demonstrated the utility of APPSHOT-generated data through a use case on radio network slicing.

Chapter 4

GenDT

4.1 Introduction

In this chapter we focus network data generation at physical layer. The conventional method to get physical layer KPIs in mobile network, or radio KPIs, is drive testing. Drive testing has traditionally been an integral part of operating mobile networks [37, 48, 144]. A key aim of drive testing is measurement based assessment and optimization of mobile network coverage, capacity and quality of service (QoS). It involves collecting field measurements in a controlled manner by driving or walking in a target scenario. Several measurement tools are available to perform drive or walk testing [8, 77, 86, 151]. The principal concern with traditional drive testing is that it requires manual effort to obtain measurements and so is costly and time-consuming.

There exist broadly two alternative approaches to reduce drive testing cost. One approach, generally referred to as Virtual Drive Testing (VDT) [26], is aimed at enabling device or infrastructure equipment testing in the lab under realistic conditions. The idea is to initially obtain a set of field measurements, as in traditional drive testing, and then recreate the field environment in the lab by replaying drive test scenarios and replicating field-measured channel conditions through a hardware channel emulator. Keysight VDT toolset [122] and Spirent Live2Lab [124] represent this approach. This approach is obviously limited to device/equipment testing and so does not cater to the needs of optimizing operational mobile networks – the latter is our focus in this paper.

The other existing approach seeks to leverage measurements from real end-user devices. From a network/operator perspective, 3GPP has introduced minimization of drive tests (MDT) feature in Release 10 to obtain measurements from actual user devices and enhanced it since [6, 84]. While this is an appealing approach and has been the focus of some industry solutions and trials (e.g., [63, 127, 200]), users' consent is needed for their devices to participate in the MDT framework, especially to provide device side context information (e.g., location) to annotate measurements. This in turn causes the issue of sparse or skewed measurement data with MDT [158]. On the other hand, inferring device locations on the network side suffers from inaccuracy along with the additional concern due to device diversity [164].

Alternatively, device side measurements can also be collected in a crowdsourced manner via dedicated measurement apps or SDKs (from third-party mobile analytics companies) installed on user devices (e.g., OpenSignal [123], Tutela [125]). The scope

and granularity of measurements that can be gathered with such crowdsourced solutions are limited by device OS APIs (e.g., Android Telephony API [75]) and so they are mostly limited to coverage mapping based on signal strength measurements [10, 51]. Crucially, the effectiveness of both MDT and crowdsourcing based measurement approaches are limited by the ability to provide incentives for users to participate and to safeguard their privacy.

Traditional generation techniques like the Taylor expansion [4] fall short in accurately producing high-fidelity time series, particularly when dealing with the actual radio signal, which can exhibit highly bursty characteristics. The Taylor expansion, while useful in various contexts, lacks efficiency in representing time series data that display such erratic and burst-like patterns. This limitation stems from the intrinsic nature of the Taylor expansion, which is more suited to smooth, continuous functions rather than the abrupt and unpredictable fluctuations typical of bursty radio signals. Therefore, for the purpose of replicating these complex, high-variance time series, alternative methods that can better capture the unique properties of bursty signals are required.

In this paper, we introduce a new approach, termed GENDT, that is powered by deep generative modeling for making drive testing efficient. Unlike the VDT approach [26, 122, 124], we design GENDT with measurement and optimization of operational mobile networks in mind. The essential idea behind our approach in GENDT is to develop a deep generative model that effectively *mimics* drive testing. Traditional drive testing results in a time-series of measurements for different radio network KPIs (e.g., RSRP, RSRQ) over a specified measurement trajectory. Similarly, GENDT takes a trajectory as an input and generates the time-series data for multiple radio network KPIs corresponding to that trajectory (see Figure 4.5 for an illustration). Note that trajectory here means a sequence of (location, timestamp) tuples so the user/device mobility is implicitly captured by this notion of trajectory. As our aim is to reduce the number of measurements required with drive testing, we use readily available network and environment ‘context’ as an aid, and train GENDT to learn the relationship between the relevant context around a measurement trajectory and the corresponding radio KPI time-series data. For the network context, we use cell site location and configuration information that an operator would hold. Points of interest (PoIs) and types of land use around the device location make up our environment context.

Given the above, the core technical problem we target with GENDT is conditional multivariate time-series data generation, where the drive testing trajectory and its context make up the condition (input) to the model to steer the data generation process, and the output is the time-series data for multiple variables (i.e., radio KPIs of interest). For training the GENDT model, we leverage a small number of controlled radio network measurements for different measurement scenarios (highway, city center, etc.) collected as with traditional drive testing. Each of these measurements is annotated with the device location and the corresponding contextual information. The GENDT model once trained as above can then be relied on to generate radio network KPI time-series data for a new unseen drive test trajectory *without* having to collect field measurements, by simply providing the trajectory and its surrounding context as input to the model.

Realizing the GENDT approach as outlined above poses a significant challenge. On one hand, GENDT should be able to generate high-fidelity (dependable) KPI time-

series data for new unseen trajectories (i.e., generalize well). On the other hand, GENDT should rely on minimal amount of measurement data for training. Addressing this challenge entails tackling a number of issues in turn: (i) *Dynamic context input*: the relevant context keeps changing as the device moves along the drive testing trajectory. This includes not only the immediate environment but also the number and the actual set of potential serving cells around the device location; (ii) *Long and complex scenarios*: drive testing trajectories can be arbitrarily long which means the model should be able to generate correspondingly long time series of radio KPIs without loss of fidelity. Moreover, real-world drive testing trajectories can be complex spanning several different measurement scenarios (highway, city center, etc.); (iii) *Stochasticity*: radio network KPIs are inherently stochastic and so the generated data should preserve this characteristic by having the distribution of synthesized data aligning with real measurement data; (iv) *Minimal training data*: the model should provide insights to optimize the amount of training data needed while ensuring high fidelity so as to strike the right balance between dependability and measurement efficiency.

In GENDT, we address (i) via a tailored Graph Neural Network (GNN) [13] based LSTM network component, where a node level network is used to map the time-varying cell information context into a high-dimensional graph; this then feeds into another aggregation network to learn the graph level information and output a multichannel time-series output, where each channel of the output represents a different radio network KPI. We tackle (ii) with a batch generation mechanism – the training and generation is done at a smaller batch level to preserve temporal patterns and improved training efficiency. We address (iii) by introducing a stochastic layer in the LSTM network and adversarial training for effectively modeling the stochastic nature of radio KPIs. Finally to address (iv), we incorporate a residual generation component in the model whose parameters give hints on model versus data uncertainty, thereby help achieve high fidelity with minimal training data.

We evaluate the GENDT with respect to a range of baseline approaches, using two real-world drive testing measurement datasets from two different countries. We not only assess the fidelity of the data generated with GENDT relative to baselines but also highlight its ability to achieve high fidelity with minimal amount of training data – the latter translates to greater measurement efficiency to benefit drive testing. All our evaluations are over the testing subset of each of the datasets that is non-overlapping with the part used for training. As such, we demonstrate the ability of GENDT to generalize to new unseen trajectories. We also present evaluations showing the effectiveness of GENDT in supporting downstream use cases as well as an ablation study to evaluate design choices underlying GENDT. In summary, we make the following key contributions:

- (§4.3) We first present an analysis of drive testing measurement data characteristics that motivate our model design.
- (§4.4) We propose a novel conditional deep generative model, GENDT, featuring several new innovations. To the best of our knowledge, GENDT is the first method for synthesizing dependable radio KPI time series data and as such the first step towards enabling efficient drive testing via generative modeling.
- (§4.6.1) Using real-world drive testing measurement datasets from two countries, we show that GENDT synthesizes realistic time series for multiple key

radio network KPIs for new unseen trajectories and generally outperforms all baselines.

- (§4.7.2) Crucially, we demonstrate the potential of GENDT to reduce the measurement effort with drive testing by leveraging the model uncertainty measure within GENDT– it maintains high fidelity for long and complex trajectories using as little as 10% of the available data, or equivalently yield 90% measurement efficiency.
- (§4.8.2) Moreover, we demonstrate the utility of GENDT for downstream applications through two distinct use cases, showing that using data generated by GENDT yields results comparable to those obtained using real drive test measurements.

4.2 Background on Device Side Measurement of Radio Networks

4.2.1 Representative Radio Network KPIs

Drive testing involves measuring a number of different radio network KPIs. Here we outline a representative set of key LTE radio network KPIs [154] that we target in GENDT.

Reference Signal Received Power (RSRP) is the average power received from a single reference signal. It typically ranges between -44 dBm (good) and -140 dBm (bad). RSRP is related to another KPI called Received Signal Strength Indicator (RSSI), which represents the total received power from the serving cell, co-channel cells and other sources of noise:

$$RSRP(dBm) = RSSI(dBm) - 10 \times \log(12^{N_{RB}})$$

where N_{RB} is the number of resource blocks.

Reference Signal Received Quality (RSRQ) indicates the quality of the received signal and typically ranges from -19.5dB (bad) to -3dB (good). RSRQ is related to the above mentioned KPIs, as follows:

$$RSRQ(dB) = N_{RB} \left(RSRP(dBm) \div RSSI(dBm) \right)$$

Based on the above, given any two of RSRP, RSRQ and RSSI, we can obtain the third. We focus on RSRP and RSRQ given their central role in influencing handover decisions for mobility management [150].

Signal to Interference plus Noise Ratio (SINR) is a key determinant of the received data rate. It is related to the transmit power, pathloss and interference.

Channel Quality Indicator (CQI) is a key KPI that is related to SINR, and is used for downlink resource scheduling and link adaptation, including the choice of modulation and coding scheme [38]. It takes discrete values between 1 and 15.

Although the above set of KPIs are a subset of KPIs considered for drive testing measurements [6], they are an essential subset as discussed above and so are sufficient

	Walk	Bus	Tram
Time Granularity	1s	1s	1s
Avg. Velocity (m/s)	1.4	5.6	11.5
Avg. Duration at each Serving Cell (s)	80.5	49.5	43.42
Avg. RSRP (dBm)	-86.6	-87.3	-85.6
Std. RSRP (dBm)	9.9	10.7	10.0
Avg. RSRQ (dB)	-14.4	-12.9	-13.3
Std. RSRQ (dB)	2.1	2.2	2.1
Measurement Samples (s)	15245	13890	14198

Table 4.1: Statistics of DATASET A for different scenarios.

	City Driving 1	City Driving 2	Highway 1	Highway 2
Time Granularity	3.8	3.5	2.1	2.3
Avg. Velocity (m/s)	9.1	9.8	26.7	31.1
Avg. Duration at each Serving Cell (s)	31.4	27.3	22.0	22.2
Avg. RSRP (dBm)	-84.6	-85.0	-86.5	-84.1
Std. RSRP (dBm)	8.8	7.1	10.5	10.2
ROC RSRP (dBm)	0.95	0.83	1.11	1.03
Avg. RSRQ (dB)	-9.5	-10.6	-8.7	-8.5
Std. RSRQ (dB)	2.0	2.5	2.2	1.9
ROC RSRQ (dB)	0.36	0.41	0.38	0.31
Sample Num.	2.1×10^4	2.3×10^4	3.9×10^4	4.6×10^4

Table 4.2: Statistics of DATASET B for different scenarios.

to highlight the potential of the proposed GENDT approach. We leave the extension of GENDT to cover additional KPIs for future work.

4.2.2 Measurement and Context Data

For our analysis and evaluation, we use two real-world mobile network measurement datasets from two different countries, both obtained through a drive testing like process. We also compile corresponding network and environment context data from public sources.

DATASET A. We collected this dataset through first-hand measurements using Nemo Handy [76], a commercial drive testing tool, mostly in and around a city center area in country A. The Nemo Handy tool allows measurement of a comprehensive set of radio network KPIs at a consistent and fine time granularity of 1s. These measurements were obtained using a custom Samsung S20 device with Nemo Handy installed. There are other studies in the literature that have reported measurements obtained using this tool (e.g., [47, 142]). Table 4.1 provides a summary of this dataset.

4.2.3 Network Context: Cell Information

For each measurement location in the above two datasets, we treat the corresponding cell deployment information as the network context. Specifically, we consider the cell

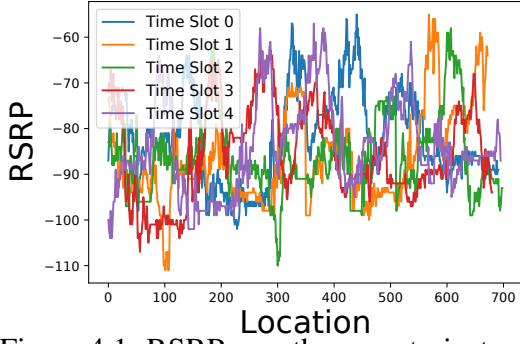


Figure 4.1: RSRP over the same trajectory with locations aligned.

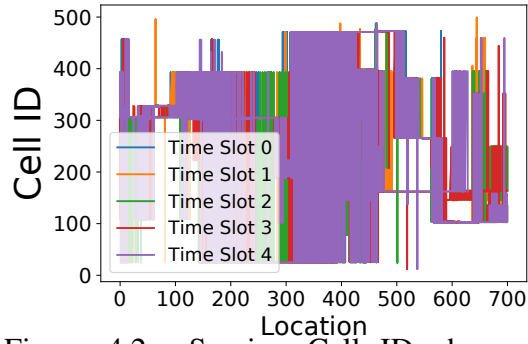


Figure 4.2: Serving Cell ID changes aligned with the RSRP in Figure 4.1.

site location, estimated transmit power and cell orientation for each cell within a certain range around the device measurement location, as such cells are seen as potential serving cells. See Figure 4.3 for an illustration¹. We discuss the setting of this range around the device in the next section. We obtain the cell site location and configuration information from CellMapper [28], a non-profit crowd sourced cell information dataset².

4.2.4 Environment Context

The radio network KPI data characteristics are not only dependent on the network context described above but also on the environment around the device (terrain, obstacles, etc.). So we additionally consider the environment context, which in our case is represented by a set of 26 attributes (see Table B.1 in Appendix B.1.1). These attributes are obtained from public sources and broadly fall into two categories: (1) land use type from Copernicus Urban Atlas repository [12]; and (2) points of interest (PoIs) from the OpenStreetMap (OSM) using the Overpass API [131]. Specifically, the value of all these attributes, centered at and within a small radius (set to $500m$ in this paper) of the device location, are taken together as the environment context. For the land use attributes, we use the percentage area of each land use type around the device as its value. For PoI attributes, we use the number of each PoI around the device as its value. Clearly, like the network context, the environment context also changes with the device location. Considering the fact that there are tens of different contexts, and that these contexts could overlap, the sheer number of possible combinations makes per-environment training infeasible. It is difficult to obtain sufficient data for all possible cases .

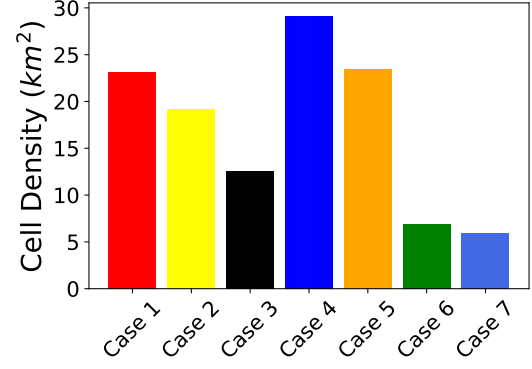
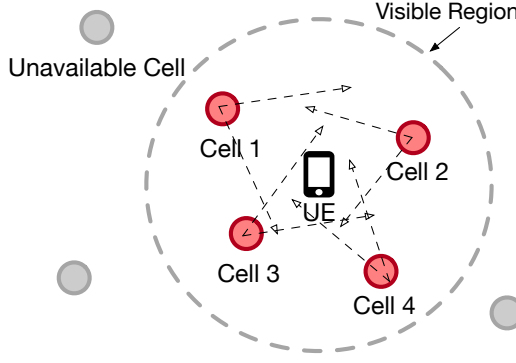


Figure 4.3: Cells in sight of a device location. Figure 4.4: Cells density in Km^2 of different cases.

4.3 Analysis of Data Characteristics

Here we present a short analysis of drive test measurement data characteristics pertinent to our model design in §4.4.

Stochasticity of radio network KPI data. Figure 4.1 shows five measurements of RSRP time series taken over the same trajectory on the tram in DATASET A around the same time and on the same day. Measurement locations are aligned across the different time series. We see significant variations between the measurements at most locations. This shows that radio network KPI data is far from deterministic, which motivates the need for a generative model capable of modeling this stochasticity as opposed to using prediction/regression models. The high level of variation of a radio KPI (RSRP in this case) at any given location is partly due to serving cell changes. Figure 4.2 shows the serving cell ID corresponding to the measurement data in Figure 4.1. We observe that in locations with high degree of RSRP variations, there are also a wide range of serving cells. This suggests that assumption of serving cell at a given location is fixed and known made in prior work (e.g., [10, 164]) does not hold in practice.

Distance to Serving Cell. From Figure B.2, we observe that distributions of distance to primary serving cell are as per intuition – slow mobility (e.g., walking) or inner city (e.g., city center cases in DATASET B) have serving cells that are relatively closer. Yet, there is considerable degree of variation in distance to serving cells within and across scenarios. A direct implication of this observation for our purpose of generating radio KPI time series data conditioned on relevant context is that the scope of the cell information context should reflect this wide diversity in order to be effective across different scenarios. For some of the scenarios, we also observe a substantial percentage of cells within almost zero distance from the device location. This reflects a common phenomenon in dense city center areas where users may pass by cells within a few meters distance and there may also be multiple cells that a user device could associate with. Also note that these plots show only the 2D distance between the device

¹Here arrows indicate the sector and direction of each cell, i.e., each cell covers the direction between two arrows ($< 180^\circ$). Dashed circle shows the furthest distance of a serving cell from the device. Cells within that range are shown in red circles. Unavailable cells beyond that range are shown as grey circles.

²Note that in practice, this information would be directly available to an operator employing our GENDT approach.

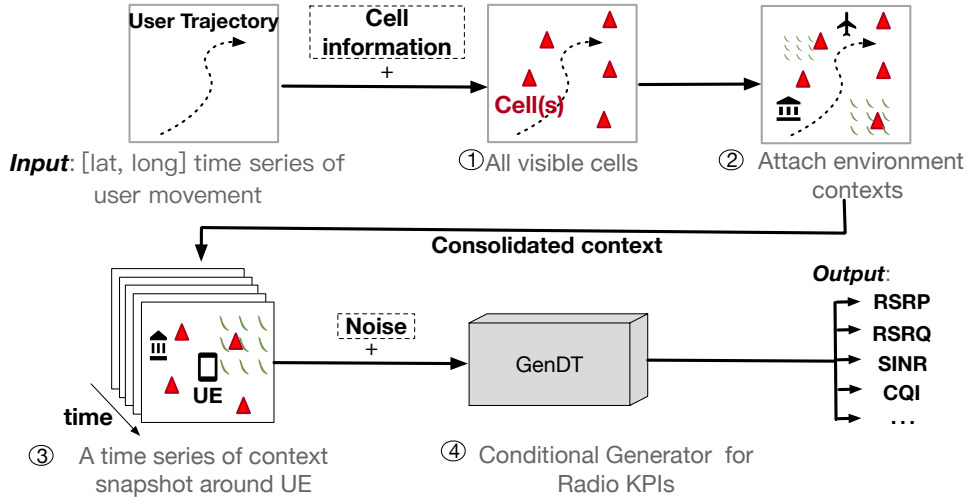


Figure 4.5: Schematic of the GENDT approach for generation of drive testing data. The whole pipeline starts from a input trajectory, represent by a series of $[Longitude, Latitude]$ with corresponding time stamp. This trajectory is then concatenate with context vector and cell information around for each time stamp, between step 1 to 3. Then the concatenated data is input to GenDT, where those information is used to do inference on the generator to get various radio KPIs.

and cell site locations.

Figure 4.4 shows the cell density differences across different scenarios³ These results also follow intuition indicating that high mobility scenarios tend to experience lower density of cells compared to inner city and slow mobility cases. Significant diversity across scenarios highlighted in Figure B.2 and Figure 4.4 emphasize the difficulty associated with generating radio KPI data for all scenarios using a single model. Though challenging, addressing this challenge is important to make drive testing efficient considering that trajectories of interest in practice span multiple different scenarios.

4.4 GENDT

4.4.1 Problem Statement

As stated at the outset, we aim at faithfully mimicking drive testing through a data generation model to reduce the need for collection of field measurements. This goal translates to generating time-series data for different radio KPIs corresponding to an input drive testing trajectory, as would be the case with traditional drive testing. Figure 4.5 illustrates the problem we target and our proposed approach to resolve it through the GENDT model. Note that this schematic depicts the operational process once the GENDT model is trained; we discuss the model training aspect shortly. The process starts with providing an input trajectory (Figure 4.5: **Input**), which is a timestamped sequence of locations for the user device (represented in $[Latitude, Longitude]$ format). Then the network context (Figure 4.5: ① as described in §4.2.3) and environment context (Figure 4.5: ② as described in §4.2.4) corresponding to each timestamp in the

³DATASET A: Case 1 – Walking, Case 2 – Bus, Case 3 – Tram; DATASET B: Case 4 – City Center 1, Case 5 – City Center 2, Case 6 – Highway 1, Case 7 – Highway 2.

input trajectory are consolidated into a series of context snapshots (Figure 4.5:③), each including the user device (UE) location at the snapshot’s timestamp. This context annotated trajectory together with noise makes up the input to the *trained* GENDT data generation model ((Figure 4.5:④), which outputs time-series data for different radio KPIs. Here context (Figure 4.5:③) serves as conditioning input to the generator, whereas noise represents factors unaccounted for in the context for the data generation process such as cell load as well as statistical variation. In the training phase that precedes the generation/operational phase outlined above, the model is trained using a small set of real drive testing measurement data. The training follows the same pipeline as in Figure 4.5 except that the model is updated based on the divergence between real and generated data.

Resolving the above outlined problem for high-fidelity and generalizable radio KPI time-series data synthesis with minimal training data is a significant challenge. A number of issues have to be addressed as part of tackling this challenge: (1) context input varies over time with device location; (2) drive testing trajectories can be arbitrarily long and complex spanning multiple different scenarios (city center, highway, etc.); (3) considering the inherent stochasticity of the radio KPI data, generated KPI data should match the distribution of the real data; (4) all of the above needs to be achieved with minimal amount of training data to achieve our intended goal of efficient drive testing.

4.4.2 Overview of Proposed Solution

Motivated by the above, we propose an original conditional deep generative model, GENDT, that addresses the aforementioned challenge and issues. Specifically, the issue (1) is addressed via a tailored GNN based time-series model, together with customized data processing, training method, and hyper-parameter tuning, as elaborated in this and the next subsection. Broadly speaking, the generation of time series data for different radio KPIs in GENDT is done in two steps, as elaborated in §4.4.3. The first step generation is conditioned on the network context (cell information). Then the environment effect is added on through a *residual* generator component (§4.4.4). We address (2) through batch training and generation (§4.4.5) that enables effective long time-series generation and training efficiency. We tackle (3) through a combination of mechanisms: noise in the input, adding stochastic layers in the different neural network components of the generator (§4.4.6) and through adversarial training (à la GANs). To address issue (4), we leverage the learned parameters of the residual generator model, whose variation offers insight on the extent to which additional training data will help improve model fidelity.

Formally, the target output of our generation model is to generate time-series data for N_{ch} different radio KPIs (e.g., RSRP, RSRQ) over a given time period T : $x'_{1:T,i} = [x'_1, \dots, x'_T]_i \in \mathbb{R}^T, i \in [1, \dots, N_{ch}]$. Here N_{ch} can be viewed as different ‘channels’ of the model output. The generated series $x'_{1:T,i}$ should exhibit high fidelity with respect to the corresponding true series: $x_{1:T,i} = [x_1, \dots, x_T]_i \in \mathbb{R}^T, i \in [1, \dots, N_{ch}]$. The whole multivariate time series data $x'_{1:T,i}$ can be generated in one shot but at the risk of compromising fidelity, especially when T is long. So we employ generation in smaller batches, each of length L . As such, the generated series can be seen as a sequence of $\lfloor \frac{T}{L} \rfloor$ batches.

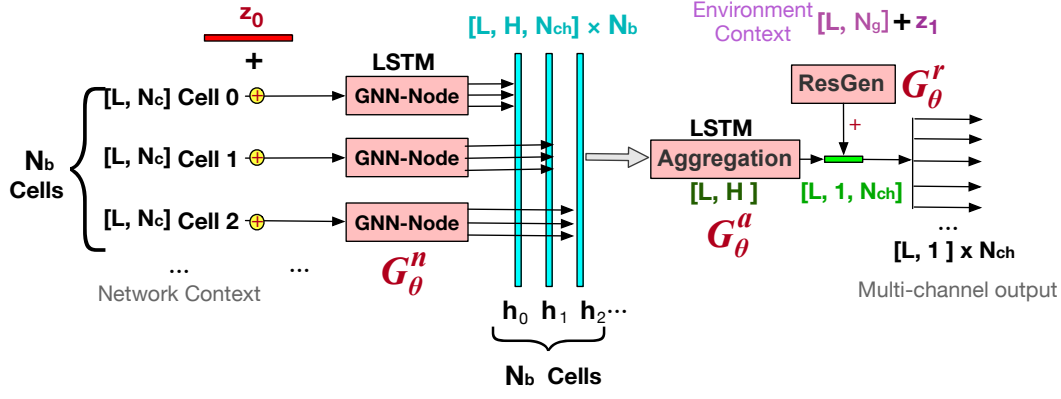


Figure 4.6: Schematic of GENDT generator architecture, labeled with input and output dimensions for each component.

The above data generation is conditioned on context c . As such, c serves as an input to the model. As noted earlier, overall context c is made up of network and environment context. The network context in each batch b is dependent on the set of potential serving (visible) cells over the course of the batch’s duration (i.e., L). As per the analysis in §4.3, we consider cells within a certain distance d_s of the user location as the relevant network context. The value of d_s is dependent on the scenario. For example, in DATASET B, we find that serving cells are within 2 km’s within the city and within 4 km’s on highways. We note that empirically and conservatively setting d_s to a higher value is sufficient for GENDT, although an unnecessarily high value increases the computation time for training.

We use $\mathcal{C}_{\text{cell},b}$ (N_b) to denote the set (number) of cells considered for the network context in a particular batch b . Note that by considering the set of potential serving cells instead of a specific one, we account for the fact that serving cells keep changing over time, as observed in §4.3. For each cell i in the set $\mathcal{C}_{\text{cell},b}$, we consider N_c attributes. In this paper, we specifically consider $N_c=5$ attributes per cell: $c_{\text{cell},i,b} = [\text{lat}_i, \text{lon}_i, p_{\text{max},i}, \text{direction}_i, \text{distance}_{i,t}]$. Here the first four are as previously described in §4.2.3. Specifically, lat_i and lon_i refer to the location of cell i , whereas $p_{\text{max},i}$ and direction_i respectively refer to the max transmit power and direction of cell i . The $\text{distance}_{i,t}$ represents the distance to cell i from the user location in time stamp t . By using this distance attribute, we implicitly account for the time-varying device location. Based on the above, the network context information in batch b is $\mathcal{C}_{\text{cell},b} = \{c_{\text{cell},i,b}\}$, $c_{\text{cell},i,b} \in \mathbb{R}^{L \times N_c}$ and $i = 1, \dots, N_b$.

Besides the network context, we also consider the environment context as described earlier in §4.2.4. Specifically, we denote the environment context in batch b using $c_{\text{env},b} \in \mathbb{R}^{L \times N_g}$, where N_g ($= 26$ in our case) represents the number of attributes considered for the environment context. Based on the above, the overall input context to our model for each batch b is $c_b = \{\mathcal{C}_{\text{cell},b}, c_{\text{env},b}\}$.

We take a data-driven approach, and accordingly design a parametric model $p_\theta(x_{1:T}|c)$ with parameter θ and fit the model on training data $\mathcal{D}$. Specifically, given training data consisting of ground-truth multi-KPI time series from M drive test measurements, i.e., $\mathcal{D} = [x_1^k, \dots, x_T^k]_i \in \mathbb{R}^T, i \in [1, \dots, N_{\text{ch}}], k \in [1, \dots, M]$, and corresponding context data c , we fit θ on $\mathcal{D}$ by finding θ^* that minimizes the divergence D between the data distribution $p_{\mathcal{D}}$ and the model p_θ , i.e., $\theta^* = \arg \min_\theta D(p_{\mathcal{D}}, p_\theta)$. Depending on the

specific training methods, different divergence criteria (D) can be considered. Once trained, we can draw samples from the model p_θ for a new target trajectory n with context c^n as input to generate the data $x'_{1:T,i}$ for that trajectory, as illustrated in Figure 4.5. Note that the training and generation process in GENDT is actually done at the batch level as outlined above and elaborated later in §4.4.5. Also note that although real world scenario characteristics can be quite different from one another (e.g., cell density differences shown in §4.3) and a target trajectory may span multiple different scenarios, our model does not need to explicitly consider the myriad of possible scenarios. This allows us to use one single model for any scenario(s).

4.4.3 Generator

As illustrated in Figure 4.6, our conditional neural sampler p_θ has three main neural network components: 1) a GNN node network G_θ^n that does convolution operation over network context (cell level information) time series; 2) an aggregation network G_θ^a to process the temporal graph after the convolution; 3) a residual generator (RESGEN) G_θ^r that accounts for the environmental effects to model the ‘residual’ and adds it to the output of the aggregation network. All these three components operate at the batch level.

- $G_\theta^n : \mathbb{R}^{L \times (N_c + N_{z_0}) \times 1} \rightarrow \mathbb{R}^{L \times H \times N_{ch}}$, where N_{z_0} is the dimension of the input noise and N_{ch} is number of target KPIs. We use a multi-channel LSTM for generation of multiple KPI time series, all together. To make sure the GNN node LSTM network does not have a bottleneck effect, we set the hidden dimension size $H \gg N_c$. Based on our empirical insights, we set $H = 100$, which we find to achieve the right balance between convergence efficiency and training performance. The additive input noise z_0 on the GNN-node network is not for introducing statistical variation but rather to help the model learn a de-noise behavior and avoid over-fitting [173]; this eases the training process and makes it robust.
- $G_\theta^a : \mathbb{R}^{L \times H \times N_{ch}} \rightarrow \mathbb{R}^{L \times 1 \times N_{ch}}$. The input h_{avg} to G_θ^a , is the high dimensional representation of the input graph. We take the average of the hidden representation of all cells as the input graph level representation, i.e., $h_{avg} = \frac{\sum_{i=1}^{N_b} h_i}{N_b}$. The aggregation network has the similar structure as the GNN-node network. Both are based on LSTM and only differ in dimensions and number of input-output channels.
- $G_\theta^r : \mathbb{R}^{L \times (N_g + N_{z_1})} \rightarrow \mathbb{R}^{L \times 1 \times N_{ch}}$, where N_{z_1} is the dimension of the input noise. The N_g environment context attributes are concatenated with the noise as input. The output of G_θ^r has the same dimensions as G_θ^a as they are added together to produce the generator’s final output. This component is elaborated further in §4.4.4.

4.4.4 RESGEN

The network context driving the first two components GENDT generator architecture (Figure 4.6) helps model the effect of cell deployment and configuration on radio net-

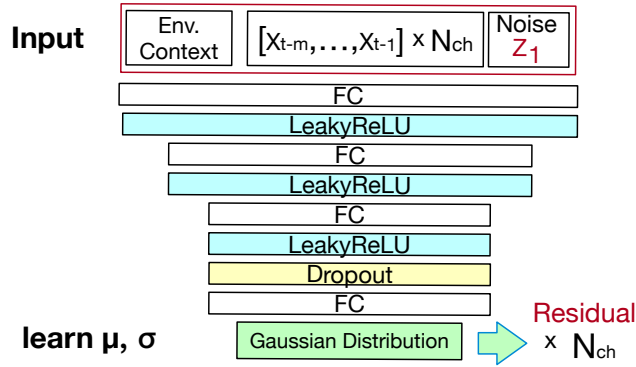


Figure 4.7: Illustration of RESGEN network architecture and the generation of distribution parameters (FC: Fully Connected Layer, LeakyReLU: Leaky Rectified Linear Unit).

work KPI dynamics but that by itself is insufficient. Environment (terrain, obstacles, etc.) has an equally important effect on radio KPI behavior. Crucially, the complexity of the environment determines the cost of drive testing (required number of measurements) in practice, as previously noted in [164]. So we design the third component of GENDT generator G_{θ}^r termed RESGEN (Figure 4.7) to model the environment effect, and crucially also to get cues on the need for additional training data. RESGEN complements the other two components in that its output (referred to as ‘residual’) is added to the output of the aggregation network to generate the final output time-series data for the target radio KPIs.

In RESGEN, we model the residual for each timestamp with a parametric Gaussian distribution, conditioned on the environment context ($c_{env,t} \in \mathbb{R}^{1 \times N_g}$), noise z_1 and the recent values of radio KPI time-series data. The latter is real (generated) data during training (generation) phase of GENDT, and importantly makes RESGEN an auto-regressive model with temporal pattern learning capability [42]. The noise input is sampled from a standard Gaussian distribution to represent the unaccounted contextual information and also for capturing statistical variation. We observe that simply using a noise input is insufficient to model the required variation on the output. Hence, we use a dropout layer [58] before the final layer of RESGEN. Once trained, we sample the Gaussian distribution $N(\mu_{\theta,t}, \sigma_{\theta,t})$ to obtain the residual, where mean μ_{θ} and standard deviation σ_{θ} are the learned distribution parameters.

Characteristics of the parameters $[\mu_{\theta}, \sigma_{\theta}]$ can be leveraged to guide the training process. They allow distinguishing between ‘model uncertainty’ and ‘data uncertainty’. If the parameters $[\mu_{\theta}, \sigma_{\theta}]$ themselves exhibit a high degree of variation during the training process, then that suggests model uncertainty and the need for more training data to stabilize these parameters. On the other hand, if the σ_{θ} has a stable but large value then that indicates that the underlying data being modeled itself has a high degree of variation and so model is not the limitation. Our target is to reduce the model uncertainty using minimal amount of training data and accordingly we leverage the above insight to that end.

4.4.5 Batch Training and Generation

In GENDT, instead of handling the whole radio KPI time series from training input or target output all in one shot, we do that in small steps called batches. We employ such

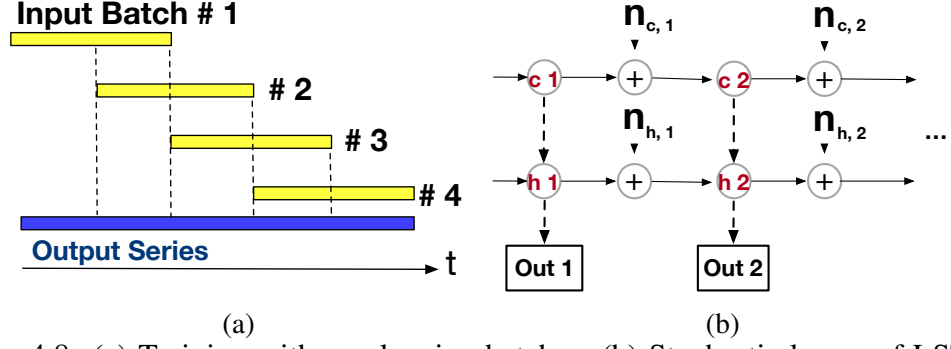


Figure 4.8: (a) Training with overlapping batches; (b) Stochastic layers of LSTM in GNN-Node Network and Aggregation Network components of GENDT.

a batch based training and generation approach for the following reasons:

- **Long series generation:** The time series of radio KPI measurements with drive testing can be quite long. We thus need to be able to generate similarly long time series but doing that in one shot risks fidelity. It is known that learning to generate long time series data at high fidelity with recurrent neural networks (RNNs), including its widely used LSTM variant, is hard [98]. So we turn the learning task of synthesizing arbitrary length series into two sub-tasks that are easier to be handle with a LSTM-based architecture: 1) learning short-term temporal correlations within each batch; 2) capturing long-term temporal correlations across batches.
- **Training efficiency:** With conditional generative models, operating at the batch level has a weight-sharing effect among batches and so enhances learning efficiency.
- **Computational efficiency:** With batch training and generation, we only need to consider context input at the batch level, which makes the processing of input more efficient compared to treating the whole time series at once.

Concretely, we view the whole training input and target output time series for each KPI as a sequence of batches, each of length L :

$$x_{1:T} \longrightarrow \{x_{1:1+L}, x_{1+\Delta t:1+\Delta t+L}, \dots, x_{1+\lfloor \frac{T}{L} \rfloor \Delta t:T}\}$$

where Δt is the step length of the sliding window, which allows different forms of batching. During the training phase, we allow the batches to be overlapping (as illustrated in Figure 4.8a) to additionally optimize the training efficiency. On the other hand, for generation, we use non-overlapping batches (i.e., $\Delta t = L$) to ensure that there are no smoothing artifacts introduced in the output and that the desired statistical variation is not compromised.

4.4.6 Stochastic Layers

The inherently and highly stochastic nature of radio KPI data (even at the same location) needs special attention to model this characteristic, especially in the generator part driven by the dynamic network context. We find that straightforward approaches to introducing noise such as injecting noise directly in the input or using a FiLM layer [136]

are ineffective in our setting. So we employ a variant of the Stochastic RNN (SRNN) method [57] to efficiently propagate uncertainty in a latent state representation with RNNs. Specifically, we use stochastic layers in the LSTM structures of both GNN-node and aggregation networks. As illustrated in Figure 4.8b⁴, we introduce noise to memories (c_t) and hidden states (h_t), where the noise is added just before each iteration. The noise modulated versions of hidden state and memory are respectively $h'_t = f_n(h_t, n_{t,h}, a_h)$ and $c'_t = f_n(c_t, n_{t,c}, a_c)$, where f_n is a function to control the intensity of noise input, and the intensity of noise added to hidden state h and c are controlled by a_h and a_c , respectively. We assume that the noise has a uniform distribution between $[0, \hat{h}_t]$ and $[0, \hat{c}_t]$, where $\hat{h}_t$ and $\hat{c}_t$ represent the average value of h_t and c_t of all hidden dimensions, so that the noise adapts to the hidden state values. Unlike the variational inference based learning used in [57], we use an adversarial training method with a discriminator. See Appendix B.1.2 for further details.

Training Following the standard GAN formulations [73], we train the model by minimizing Jensen-Shannon divergence, *i.e.*, $\theta^* = \arg \min_{\theta} JS[p_{\mathcal{D}}||p_{\theta}]$, and with the aid of discriminator as in the GAN framework. We denote such discriminator as R due to their role as density ratio estimators [172]. Specifically, for given training input measurement data time series and context batch $(\underline{x}, \underline{c})$, the corresponding adversarial loss between the data $p_{\mathcal{D}}(\underline{x}, \underline{c})$ distribution and the model $p_{\theta}(\underline{x}, \underline{c})$ distribution is defined as:

$$\mathcal{L}_{JS}^R(p_{\mathcal{D}}, p_{\theta}) = E_{p_{\mathcal{D}}}[\log R(\underline{x}, \underline{c})] + E_{p_{\theta}}[\log(1 - R(\underline{x}', \underline{c}))].$$

In our case, we consider one discriminator, named as R_{θ} , the context input into discriminator is the high dimensional representation of c , which is h_{avg} . The discriminator is a single layer LSTM network.

We additionally use the standard mean squared error loss:

$$\mathcal{L}^M(\underline{x}, \underline{x}'|\underline{c}) = \frac{1}{L} \sum_{t=1}^L (x_t - x'_t)^2$$

Since the batch length L is constant during training, this loss has an equivalent effect to using $L2$ loss.

Overall, together with the adversarial (GAN) loss, the loss function to fit θ is:

$$\mathcal{L} = \mathcal{L}^M + \lambda \mathcal{L}_{JS}^R$$

where the λ is a weight to balance the effect of adversarial loss, which in our case is set as $\lambda = 0.1$ by default. Appendix B.1.3 elaborates on the setting of hyperparameters.

4.5 Evaluation Methodology

Broadly speaking, we evaluate GENDT in two ways. First, we assess the fidelity of the GENDT generated radio KPI time series data with respect to real measurement data using multiple different metrics described in §4.5.1 and in comparison with various baseline approaches outlined in §4.5.2. Second, we evaluate GENDT through two

⁴Here we show for one radio KPI (channel) case but the same applies for all channels.

different downstream use cases and show that GENDT generated data is a dependable substitute for real drive testing measurement data to support such use cases. When split the training set, validating set, and evaluation set, we make sure there is no overlapping regions or trajectories between those three set by a longitude-latitude filter, therefore there is no overlapping between training set and evaluation set.

4.5.1 Metrics

Mean Absolute Error (MAE) for any given KPI between its real measurement data time series ($x : \{x_1, x_2, \dots, x_T\}$) and generated time series ($y : \{y_1, y_2, \dots, y_T\}$) is calculated as: $MAE = \sum_{i=1}^T |y_i - x_i|/n$. As such, it is a natural measure for evaluating fidelity of GENDT and alternative approaches.

Dynamic Time Warping (DTW) [27] is an alternative metric to MAE for assessing the similarity between two time series (real and generated in our setting). The main feature of this distance measure is that it allows to recognize similar shapes between two time-series signals, even if they need signal transformations such as shifting and/or scaling. As such, it provides a more robust similarity measure. Events like accessing a specific cell or going around the same location have a similar effect on the temporal pattern of KPIs across different measurement trajectories, though with slight time shift due to differences in user device path and velocity each time. DTW is better at identifying such similarity, as the other distance metrics are too sensitive to temporal shifts. Hence, the DTW is very useful in capturing real world performance, especially when used in conjunction with MAE, as we do. **Histogram Wasserstein Distance (HWD)**. Besides having the generated time series of different radio KPIs matching with their corresponding ground-truth time series (as quantified by the MAE and DTW metrics), we would also want the generated data for any target KPI to have the same distribution (histogram) as the real data. Rather than limiting the comparison of histograms of real and generated data to just visualization, we quantify the similarity between these histograms by computing their Wasserstein Distance (WD) [148] and call this metric as the Histogram Wasserstein Distance (HWD).

Measurement Efficiency. While fidelity of the generated data along different aspects as quantified by the above metrics is important, the required amount of training data to achieve that fidelity is equally important. Lower the training data needed the better as it demonstrates the cost reduction and efficiency improvement that GENDT can provide, aligned with the motivation behind its design. As different scenarios involve different movement speeds, lengths of trajectories included in the training data in terms of distance are not representative. We therefore factor in speed in trajectories and consider data used for training in terms of time ($\sim$ distance/speed). Specifically, we use the percentage of the available data in a dataset that is used for training as our measurement efficiency metric.

4.5.2 Baselines

We are unaware of any other work in the literature adopting a generative modeling approach like ours for efficient mobile network drive testing. So we consider a range of alternative approaches from other domains as baselines.

Method	MAE↓			DTW↓			HWD↓		
	Walk	Bus	Tram	Walk	Bus	Tram	Walk	Bus	Tram
GENDT	5.17	8.88	7.49	2.4	5.2	3.6	7.2	3.7	3.9
FDaS	11.2	15.6	13.1	15.5	19.0	17.2	6.9	4.2	3.3
MLP	9.9	11.5	12.1	9.0	11.1	7.8	13.8	9.9	12.9
LSTM-GNN	21.5	18.3	12.8	12.1	14.2	15.1	9.7	12.3	12.4
Orig. DG	10.8	14.3	12.7	11.9	16.1	14.5	11.9	13.4	10.1
Real Cont. DG	9.2	10.43	7.71	5.1	7.9	5.2	12.6	11.2	4.9

Table 4.3: Generated RSRP time series fidelity with GENDT and baselines for different scenarios in DATASET A.

Fit Distribution and Sample (FDaS). FDaS [43, 128] is another simple minded baseline that focuses on modeling the distribution (histogram) of the data for any given radio KPI. Specifically, it fits a distribution based on the real KPI data (ignoring the time dimension) using maximum likelihood estimation, and samples from it afterwards to generate the data for that KPI. While this baseline can be effective with respect to the HWD metric, it can be quite poor in terms of the other fidelity metrics as it does not consider relationship with context nor the temporal relationships in the data.

Multilayer Perceptron (MLP) is a simple minded baseline that infers the data for each radio KPI independently at each time step through regression over the context input. Clearly, this baseline does not account for the temporal relationships within the real KPI time series data. Moreover, as it focuses solely on the relationship between context and KPI data, it does not model stochasticity of the latter either.

LSTM-GNN [168], a variant of [66], is a state-of-the-art model architecture for GNN based time-series prediction. We use it as a baseline as an alternative approach especially with respect to the first two neural network components of GENDT generator (§4.4.3), and highlight the benefit of GENDT’s handling of dynamic context input, batch based generation and use of stochastic layers.

DoppelGANger (DG) [98] and Variant. As mentioned in §2.5, DG is a state-of-the-art multivariate time series data generation model and so is a natural baseline approach to compare with. The original DG model (depicted in Figure B.3a) generates the context in its first stage. In our problem setting, however, this context data is readily accessible to the operator and can be directly used without having to learn to generate it. So we additionally consider an optimized variant of DG called ‘Real Context DG’ in which we bypass the context generation stage and directly input real context to the second stage time-series data generator in DG, as depicted in Figure B.3b.

4.6 Evaluation Results

Here in §4.6.1 we first the evaluate GENDT on the fidelity metrics from §4.5.1 and benchmark it against the baselines outlined in §4.5.2. Then we demonstrate that the uncertainty measure within GENDT can be used to optimize measurement efficiency (§4.7.2). In §4.8.2, we demonstrate the value of GENDT-generated data for two downstream use cases. Finally, we carry out an ablation study of GENDT to examine the effect of its underlying design choices (§B.3.2).

Method	RSRP			RSRQ			SINR			CQI		
	MAE	DTW	HWD	MAE	DTW	HWD	MAE	DTW	HWD	MAE	DTW	HWD
GENDT	7.18	3.73	4.93	1.9	1.27	13.2	4.0	4.6	7.2	1.9	1.20	3.8
FDaS	13.63	17.23	4.80	2.8	1.80	10.1	8.2	6.2	5.9	3.1	1.90	3.8
MLP	10.83	9.3	12.20	2.4	1.70	11.0	7.6	5.9	9.0	2.7	1.33	6.1
LSTM-GNN	17.53	13.80	11.47	2.8	1.81	13.1	9.6	6.9	11.2	3.0	1.55	4.1
Orig. DG	12.93	14.17	4.98	2.9	1.86	11.9	8.8	5.9	6.5	3.2	1.60	3.8
Real Cont. DG	9.11	6.07	10.2	2.2	1.69	12.5	5.3	5.4	8.5	2.1	1.25	4.3

Table 4.4: Average performance of GENDT and baselines across all scenarios in DATASET A for RSRP, RSRQ, SINR, and CQI time series generation.

Method	MAE↓				DTW↓				HWD↓			
	City C- enter 1	City C- enter 2	High- way 1	High- way 2	City C- enter 1	City C- enter 2	High- way 1	High- way 2	City C- enter 1	City C- enter 2	High- way 1	High- way 2
GENDT	4.9	4.8	8.5	8.9	2.8	2.9	5.1	5.4	3.8	1.3	5.2	7.3
FDaS	9.8	11.7	16.7	10.8	6.5	8.8	14.8	10.1	3.4	3.1	7.9	6.4
MLP	8.5	3.2	14.5	16.9	5.6	3.1	11.9	15.2	4.1	2.8	18.7	14.0
LSTM-GNN	19.7	16.8	18.3	13.6	12.1	11.8	14.2	11.2	8.6	10.0	8.5	8.0
Orig. DG	15.6	14.3	17.1	14.6	11.5	10.1	10.4	9.8	5.0	3.2	9.5	9.2
Real Cont. DG	10.3	7.4	9.1	9.4	3.9	4.6	6.0	5.9	3.8	2.9	11.8	9.8

Table 4.5: Generated RSRP time series fidelity with GENDT and baselines for different scenarios in DATASET B.

4.6.1 Fidelity and Generalization

Setup To assess the generalization capability of GENDT to new unseen trajectories, we split each of our datasets into two non-overlapping parts: training and testing. We further make sure to avoid geographic proximity between training and testing measurement data locations. *We only report performance on the testing set throughout this whole section.* While we show results of GENDT (and other baselines) in different scenarios separately to highlight the versatility of GENDT, note that these are all generated using the same GENDT model.

DATASET A. Here we present evaluation results with DATASET A focusing on generation of time series for RSRP, RSRQ, SINR and CQI KPIs. We first carry out the per scenario evaluation focusing on RSRP, before evaluating the average performance of GENDT for all KPIs across all scenarios.

By comparing the performance of different methods under multiple metrics in Table 4.3 for the generated RSRP KPI time series, we observe that the GENDT generally yields the best performance of each scenario for all metrics. Though FDaS expectedly can model the data distribution well (measured by HWD metric), its performance on other two metrics (particularly DTW) is the worst among all the alternatives compared. MLP performance is intermediate to worst on all metrics, especially in terms of HWD, as it does not model stochasticity and temporal behavior. The HWD performance of LSTM-GNN is similar to that of MLP due to the same underlying reason. Interestingly, it exhibits rather poor performance on MAE and DTW, even worse than MLP that does not model temporal variation at all. We attribute this to two reasons: (1) LSTM-GNN is a prediction model not a generative one; and (2) it does not have mechanism for effective long series generation.

The original DG model, despite being a time-series data generation model, performs poorly across all metrics, about similar or worse than MLP and LSTM-GNN. The performance of the original DG is limited by its function of utilizing the contexts. As an unconditional generative model, the original DG model does not take in any contexts but generate everything from noise input, therefore it cannot use the con-

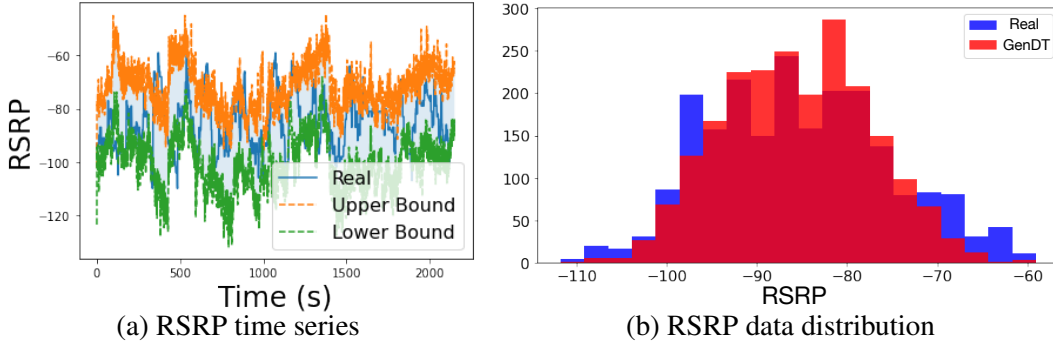


Figure 4.9: Visualization of GENDT performance evaluated over a long and complex trajectory in DATASET B.

textual information effectively. Still, it yields only intermediate performance due to its inability to handle dynamic network context input and insufficient mechanisms to capture stochasticity, latter clearly reflected in the poor HWD performance relative to GENDT. The shortcoming of real context DG relative to GENDT with respect to the former context handling issue and the effectiveness of GNN structure in GENDT to that end is illustrated in the generated RSRP series with these methods in Figure B.4 (in Appendix B.3).

Considering all the considered KPIs including RSRP, the average performance across all scenarios is reported in Table 4.4. We observe that the big performance improvements seen with GENDT above continue to hold with the exception of CQI performance, where benefits are somewhat marginal. We attribute this to the fact that, unlike other KPIs, CQI generation is a classification problem involving a choice of one among discrete values from 1 to 15. Overall, we observe that the overlapping batches based training on top of batch generation and handling time-varying relevant context input plays a key role in the superior performance of GENDT, so does the SRNN structure in the generator (§4.4.6) which helps in better modeling the data distribution.

DATASET B. We now consider DATASET B which consists of longer and more complex movement trajectories over a wider geographical region. This dataset, however, lets us evaluate with respect to generation of time series for only RSRP and RSRQ KPIs as it lacks the other KPIs.

As before, we first consider RSRP and report performance at the per-scenario level in Table 4.5. Again, we observe that GENDT generally yields the best performance and FDaS doing marginally better in terms of HWD as expected. The average performance across all scenarios is reported in Table 4.6, also considering the RSRQ KPI. We notice that relative to significant improvements seen with GENDT in the case of RSRP, gains for RSRQ are less striking. We find that this is because the RSRQ values in the test scenarios are fairly stable and also vary in a much smaller range than RSRP, thereby limiting the room for improvement.

4.6.2 Long and Complex Scenarios

We now consider a long continuous trajectory lasting 2230s (~40mins) as the testing set to evaluate GENDT and baselines for generation of long series of radio KPI data

Method	RSRP			RSRQ		
	MAE↓	DTW↓	HWD↓	MAE↓	DTW↓	HWD↓
GENDT	6.78	4.05	4.40	1.7	1.40	8.1
FDaS	12.25	10.05	5.20	2.9	1.98	10.8
MLP	10.63	8.95	9.90	2.6	1.81	8.5
LSTM-GNN	17.1	12.33	8.78	2.4	2.0	12.9
Orig. DG	17.93	9.17	11.80	2.9	1.86	12.9
Real Cont. DG	9.05	5.10	7.08	2.0	1.53	11.1

Table 4.6: Average performance of GENDT and baselines across all scenarios in DATASET B for RSRP and RSRQ generation.

Method	RSRP			RSRQ		
	MAE↓	DTW↓	HWD↓	MAE↓	DTW↓	HWD↓
GENDT	11.69	7.18	10.4	3.9	2.40	2.1
FDaS	24.25	16.05	19.20	10.8	13.1	2.98
MLP	18.63	14.95	29.90	8.61	9.9	4.6
LSTM-GNN	18.1	13.80	30.78	10.45	9.9	4.9
Orig. DG	20.40	13.45	26.73	10.1	13.9	2.3
Real Cont. DG	15.05	10.80	27.08	5.08	7.1	3.0

Table 4.7: Overall performance of GENDT and baselines for long and complex trajectory case in DATASET B.

over a complex scenario. The considered trajectory spans three cities in DATASET B (Wuppertal, Hamm, and Koln), including inner city driving and highway driving between them. The total length of the trajectory is about $40km$. We make sure that this test trajectory does not overlap nor has significant proximity to trajectories in the training set. Moreover, the training set does not include data from any of the three cities or routes between them.

We first show qualitative results in Figure 4.9, where we can see that the generated RSRP series with GENDT varies in a range that tightly covers the ground truth (Figure 4.9a), and also shows good match with ground truth in terms of RSRP data distribution (Figure 4.9b). Note that the upper/lower bounds shown in Figure 4.9a are not themselves generated time series with GENDT. Rather, they represent min/max statistics of the generated samples for each time instant. We then summarize the quantitative results in Table 4.7 that show the overall performance of GENDT compared to baselines. We see that GENDT consistently and significantly outperforms on all metrics for both RSRP and RSRQ. These results particularly highlight the benefit of batch generation given the length of the target trajectory with only Real Context DG coming close to the performance of GENDT. The additional measures in GENDT to aid in effective long series generation (autoregressive RESGEN) and beyond (GNN structure and stochastic layers) explain its superior performance. These results also highlight the pitfall of FDaS as data distribution of the complex target trajectory is not captured by the training set and so FDaS yields poor performance even in terms of HWD. We further discuss the need of long trajectory generation in §B.3.1 with quantitative results and illustrative examples.

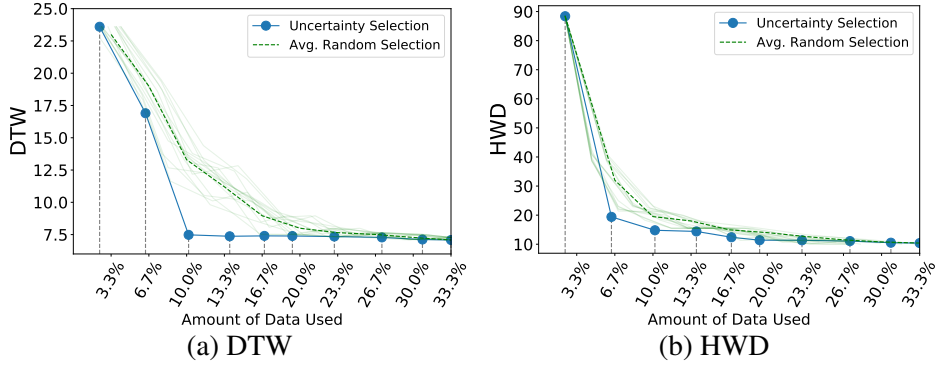


Figure 4.10: Assessing selection of new training data based on GENDT uncertainty measure relative to random selection.

4.7 Measurement Efficiency

4.7.1 Model Uncertainty

Data uncertainty is irreducible due to the nature of the data while model uncertainty can be reduced by training on more data and actively selecting new training points [58]. The design of GENDT naturally decouples data and model uncertainty: the data uncertainty is reflected by the actual value of the standard deviation in the learned Gaussian distribution from RESGEN while the model uncertainty is determined by the variation of the Gaussian parameters. We use MC dropout [58] to obtain the model uncertainty of GENDT, i.e., the dropout is turned on during generation time to obtain multiple outputs of the model. As the parameters of observation model (mean and standard deviation of the parametric Gaussian) are the (direct) output of the neural network of RESGEN, we use the standard deviation of them averaged over time as the model uncertainty. Specifically, the model uncertainty is defined as:

$$U(G_\theta) = \frac{1}{T} \sum_{t=1 \dots T} std(\sigma_\theta)_t + std(\mu_\theta)_t$$

where T is the length of target series and std is the standard deviation computed by empirical samples with dropout turned on.

4.7.2 Uncertainty Driven Measurement

We evaluate the usefulness of the model uncertainty in an active learning setup on DATASET B, mimicking a real-world uncertainty driven drive test measurement data collection process.

Here we take the long trajectory in §4.6.2 as the testing set (named as S_L). We remove the testing set from DATASET B, and split the rest of the data into 23 subsets with no overlap in geographical region between them. We initially start with just one small subset of data as the training set. At each step, we evaluate the trained model on each of the remaining subsets in the data to obtain the model uncertainty, and select the one with highest uncertainty as new training data to add to the current training set. Concurrently, we evaluate the GENDT model performance on S_L at each step to assess the benefit with the above uncertainty guided training data selection. As

shown in Figure 4.10, just after two steps (with 10% of the available data used), the performance on S_L no longer shows clear improvement on both DTW and HWD. We omit MAE results for brevity as they are similar to DTW.

As an alternative approach, we perform random selection with the same starting subset of the selected 10 subsets. In other words, we follow the same process as above but at each step randomly selecting the training point to add instead of relying on the uncertainty measure.

Results in Figure 4.10 shows that for the same number of selected subsets, the random selection always shows lower training efficiency compared to the uncertainty based method. Furthermore, the random selection never goes into a case where its performance is better than uncertainty based selection, which means that the uncertainty based method does provide an optimal path to add the most informative data. Overall, with uncertainty guided (random) training data selection, 10% (20%) of the available data (23 subsets) is sufficient to achieve the most generalization that can be evaluated for DATASET B. We could equivalently view this as achieving 90% (80%) measurement efficiency compared to traditional drive testing. Indeed, this efficiency could be higher as the model can generate many more trajectories for which ground truth may not be available.

4.8 Downstream Use Cases

In this section, we assess how well our GENDT approach can support drive testing use cases. The general idea here is to consider use cases that depend on drive testing measurement data, and evaluate the effect of using GENDT-generated data for those use cases in comparison with using actual measurement data. The choice of the use cases highlighted is constrained by the access to ground-truth radio KPI measurement data to conduct such an evaluation. In the following, we present results for two distinct use cases, each relying on data for a different set of radio KPIs. In Appendix B.3.3, we discuss further use cases that GENDT can support.

4.8.1 Mobile Service Quality of Experience (QoE) Prediction

User QoE assessment is a key focus of mobile network operators for which they engage in drive test measurement data collection. Application layer throughput is a key QoE metric of interest that in turn depends on lower layer radio KPIs such as RSRP and RSRQ [139, 140]. We also consider Packet Error Rate (PER) as another key QoE metric. We focus on DATASET A that not only includes drive/walk testing based measurement data for multiple radio KPIs collected with Nemo Handy [76] but also corresponding downlink throughput and PER measurements obtained with iPerf3 [50].

For QoE prediction, we leverage a recent work [161] that examined machine learning based prediction of application QoE metrics like throughput based on drive testing based radio KPI measurement data, including RSRP and RSRQ. In particular, we use the MLP based regression model for QoE metric prediction from [161] that uses RSRP, RSRQ, device location, etc. as features. We first confirm that RSRP and RSRQ KPIs are critical for accurate QoE prediction with this model by dropping these two KPIs from the model and observing the significant divergence between real (measured) and predicted throughput (see Figure 4.11a and second row in Table 4.8). In contrast,

Method	Throughput			PER		
	MAE↓	DTW↓	HWD↓	MAE↓	DTW↓	HWD↓
Real	6.7	4.0	1.2	0.22	0.18	1.9
RSRP & RSRQ Excluded	13.1	9.6	2.4	0.48	0.39	3.8
GENDT	5.9	4.6	1.4	0.24	0.23	2.7
FDaS	13.4	9.9	2.4	0.48	0.30	3.5
MLP	8.6	5.9	2.1	0.33	0.38	3.2
LSTM-GNN	14.0	9.4	2.5	0.35	0.39	3.4
Orig. DG	13.1	10.1	2.3	0.47	0.39	3.3
Real Cont. DG	7.9	5.1	1.2	0.28	0.31	2.8

Table 4.8: Performance with GENDT-generated RSRP and RSRQ data when applied to QoE (throughput and PER) prediction use case, relative to baselines.

including measured RSRP and RSRQ KPI data greatly improves the throughput prediction (see Figure 4.11b and first row in Table 4.8).

To assess the usefulness of GENDT for this use case, we now evaluate the effect of using GENDT-generated RSRP and RSRQ time series data. Quantitative results are shown in Table 4.8 when using data generated with GENDT and baselines. Note that we use the same fidelity metrics of MAE, DTW and HWD as before, except that these results evaluate the fidelity of predicted throughput and PER time series with respect to their real (measured) series. We observe that GENDT-generated RSRP/RSRQ data yields QoE predictions very similar to that of using corresponding real data, and much superior to using data generated with baselines.

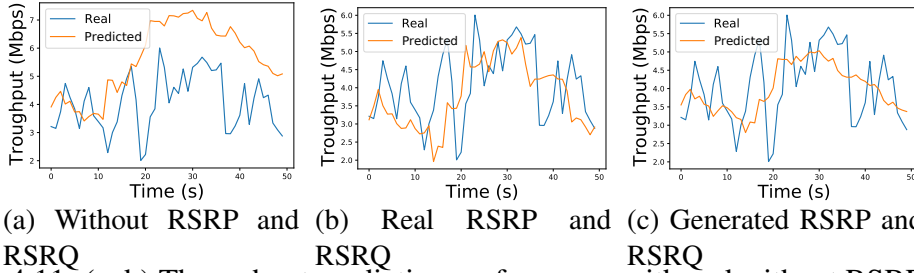


Figure 4.11: (a, b) Throughput prediction performance with and without RSRP/RSRQ KPI measurement data, and (c) with GENDT-generated RSRP/RSRQ data.

4.8.2 Analysis of Handovers

Optimizing the handover frequency and performance is of key importance to mobile network operators as too many handovers can not only degrade user experience but also increase signalling overhead in the network. This is done in practice by tuning thresholds of multiple KPIs relevant for mobility management informed by drive testing measurement data on handovers [150].

To support this use case on inferring handovers for a given network deployment, we retrained GENDT to generate the time series of an additional KPI – the serving cell. Tracking serving cell changes essentially provides the information on time between handovers. Note that GENDT model itself remains unchanged from what is described in §4.4 to accommodate this new serving cell KPI. Quantitative results from Table 4.9 clearly show that GENDT-generated serving cell data provides inter-handover time

Method	HWD↓
GENDT	2.4
FDaS	8.3
MLP	6.1
LSTM-GNN	5.3
Orig. DG	8.0
Real Cont. DG	3.0

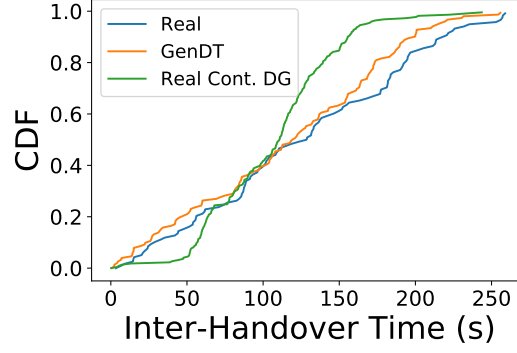


Table 4.9: Inter-handover time distribution estimation with GENDT-generated serving cell data, relative to baselines. Figure 4.12: Inter-handover time distribution from GENDT-generated serving cell data compared to real distribution in DATASET B.

distribution that closely match with real data compared to the baseline approaches. This is also apparent from the CDF of inter-handover times with GENDT shown in Figure 4.12 compared to their real counterpart from drive test measurements in DATASET B. In contrast, inter-handover times from DG-generated data are off from the real.

4.9 Discussion

4.9.1 Why generating point coverage is insufficient

The generation of data as discrete points falls short in adequately representing the complexities of user mobility. In the context of network performance, user mobility is a crucial factor influencing handover behavior and the potential for signal fading associated with varying velocities. To truly capture these dynamics, it is essential to generate radio Key Performance Indicators (KPIs) for specific user trajectories rather than relying on static point-based data. This approach acknowledges that mobility patterns are often fluid and varied, and a static, point-based model can only represent a limited scenario, such as when the user’s trajectory is stationary or has a constant value. Incorporating trajectory-based KPI generation allows for a more realistic and comprehensive understanding of network performance in the face of real-world user movement and behavior.

4.9.2 Comparison with Virtual Drive Testing

Virtual Drive Testing (VDT) serves a purpose distinct from that of real-world drive testing. VDT is designed specifically for testing network equipment under controlled conditions that simulate real-world driving scenarios. In contrast, our approach focuses on replicating the process of collecting real-world drive testing data through the use of generative modeling. This method allows us to create synthetic data that closely mirrors the kind of information gathered during actual drive tests, without the need for physical testing conditions. Our goal is to harness the power of generative models to simulate real-world data collection, offering an innovative alternative to traditional drive testing methods.

Our current resource constraints preclude the possibility of conducting large-scale VDT. Additionally, based on our understanding, VDT tends to simulate highly specific environments rather than replicating the complexity and scale of real-world scenarios. This inherent limitation in VDT’s capacity to generalize to unseen cases poses a challenge in drawing fair comparisons. Therefore, while VDT can offer insights into certain controlled environments, its application in our context may not provide a comprehensive or accurate reflection of the diverse and dynamic real-world conditions we aim to analyze. This disparity between the simulated environments of VDT and the multifaceted nature of real-world scenarios is a significant factor in our decision to not utilize VDT for large-scale testing in our study.

4.9.3 Weather and Other Critical Contexts

Ideally, our dataset should include weather conditions, such as rain, due to their substantial influence on latent variables that are often treated as noise in our current model. This is because, in the absence of specific weather information during data collection, these environmental factors remain unaccounted for, leading to a significant limitation in our dataset. Recognizing this, our proposed method has the potential to effectively capture and integrate the impact of weather conditions, provided it is trained on a sufficiently comprehensive dataset that includes detailed weather information. Incorporating such data would enhance the model’s accuracy and predictive capabilities, allowing for a more comprehensive understanding of how weather influences the parameters we are measuring. This represents a key area for improvement and expansion in future iterations of our research, underlining the importance of diverse and detailed datasets in enhancing model performance.

4.10 Summary

In this chapter, we proposed GENDT, a GNN-based generator for generating radio KPIs. We adapt GNN to leverage the context data that have graph structure with time-varying number of nodes. Stochastic RNN is used to generate the small-scale fade of radio signal, whereas we use a Gaussian noise with learned parameters to generate the large-scale variation. Meanwhile, the parameter of learned Gaussian distribution can represent the model uncertainty to a specific input, which in return guide the measurement in the real world to cover the place that can reduce the model uncertainty and accelerate the whole process of measurement. We evaluate GENDT on two city-scale datasets in two countries that include many different cities. The results show that GENDT can achieve perfect generalization with fewer than 10% training data across the entire region, with significantly better fidelity of the radio KPIs generated.

Chapter 5

DistilGAN

5.1 Introduction

Telemetry is a collection of measurements or other data at remote points and their automatic transmission to receiving equipment (telecommunication) for monitoring. Telemetry to accurately understand network behavior and manage networks is a fundamental and longstanding problem in many scenarios: from wired to wireless networks including IoT/sensor deployments. So far, while many methods have been proposed, we observe that none of them can simultaneously satisfy the four essential requirements telemetry methods must meet: (1) high-fidelity: ability to reliably capture fine-grained network status; (2) efficiency: minimizing the amount of telemetry information sent by network nodes to collectors for analysis and control; (3) versatility: being able to work in varied network telemetry settings and with diverse resource constraints; (4) real-time: having collectors receiving data from nodes capable to quickly recover fine-grained network status from the received information to support real-time network management.

In this chapter, we show that this is possible by proposing DistilGAN, a new data-driven deep generative modeling powered solution anchored at the collector in a network telemetry system that only requires sampling of raw measurement data from network nodes at the continually specified minimal rate. We show the effectiveness of DistilGAN in meeting all the above four requirements via extensive evaluation spanning three real and diverse scenarios: (1) a large ISP network; (2) 5G radio access network; and (3) IoT (smart meter) network. Compared to wide range of baselines representing the state of the art, we show that DistilGAN yields significant gains in fidelity, efficiency, robustness and real-time capabilities, across the three diverse scenarios and downstream use cases considered. Representative instances of such gains include 25% better efficiency compared to state-of-the-art method for flow-level monitoring in ISP networks for the same fidelity, and 1-2 orders of magnitude faster data stream recovery compared to state-of-the-art approaches.

5.2 Requirements for Network Telemetry Methods

5.2.1 Fidelity

The utility of network telemetry data hinges on the extent to which it represents the actual network state. So high fidelity is an essential requirement that network telemetry methods must meet. This in turn means being able to access fine-grained measurement information at the collector that reflects what is observed at the individual network elements.

5.2.2 Efficiency

This requirement arises from the fact that communication of network monitoring data to the collector needs to share the same underlying network resources with regular operational network use. So the network telemetry method must be communication efficient by keeping the monitoring overhead minimal and not causing disruption to normal network operation. This in turn implies that simply reporting all raw measurements from the network elements at their original frequency is not viable. Not only that, the telemetry must also adapt what is reported in accordance with change in data characteristics over time.

5.2.3 Versatility

Different network telemetry settings (e.g., ISP networks, data centers, cellular networks, sensor and IoT networks) differ in their telemetry KPIs and data characteristics. Equally, network elements also exhibit differences in their capabilities and resource constraints (e.g. processing power, available memory). So it is desirable for a network telemetry method to be not only suitable for a variety of telemetry settings but also to be robust in the face of resource constraints.

5.2.4 Real-time

Network telemetry method must support reconstruction of the network state at the collector from the received telemetry data streams as fast as possible so to allow quick and real-time response to routing imbalance [96, 99], network failures [118] and attacks [170, 189]. Such data reconstruction has to at least keep pace with the arrival rate of the data streams.

5.3 Motivation

Network telemetry systems are typically composed of two main components [190]: (1) network elements (routers, switches, base stations, IoT devices, etc.) from where measurement data originates and is reported; and (2) a centralized collector, where the reported data is collected together to enable operators to analyze network behavior and inform control decisions to effectively manage the network. The method employed for

Approach	Fidelity	Efficiency	Versatility	Real-time
Sampling [14, 137, 145, 167, 185, 186]	?	?	✓	✓
Sketching [70, 95, 100, 101, 182]	?	✓	×	✓
Sensing & Recovery [18, 34, 46, 69, 71, 191]	✓	?	?	×
Our approach — DISTILGAN	✓	✓	✓	✓

Table 5.1: Qualitative comparison between existing network telemetry approaches and proposed DISTILGAN.

acquiring telemetry data is central to the above outlined system and it must meet the requirements in §5.2: High Fidelity, Efficiency, Versatility, and Real-time.

5.3.1 Limitations of existing approaches

Broadly, three distinct approaches exist for network telemetry, as discussed below. Generally speaking, they make different trade-offs between the various requirements and as such none of them meet *all* the aforementioned requirements, as summarized in Table 5.1.

Sampling

Sampling is the basic and most commonly used network telemetry method. As the name suggests, the idea behind sampling is to ‘sample’ the raw measurements at the network elements and report only those samples. When applied to packets and flows in IP networks, this translates to sending raw data in the form of samples of $\langle \text{flowID}, \text{count} \rangle$ tuples [14, 137] or packets [145, 167]. This approach has minimal resource requirements at the network elements and can be applied to any network telemetry setting, thus making it versatile. In fact, sampling is typically an integral part of other network telemetry approaches discussed next. Sampled data is also easy to handle at the collector, contributing to its suitability to support real-time control.

However, the fidelity and efficiency with this approach is determined by how the sampling rate is selected. As noted in [185], the approach taken in practice is to be conservative and use the highest possible sampling rate to ensure high fidelity and not risk missing out on any important insights from the data. Clearly, this can be highly inefficient. Nyquist sampling frequency from the signal processing domain offers a structured way to pick the minimal sampling rate [130]. A telemetry KPI sampled at the Nyquist rate and put through a fourier transform on the sender side can be perfectly recovered at the receiver (collector) side with an inverse FFT operation [185, 186]. However, Nyquist rate varies with the data characteristics over time and so the sampling rate needs to be continually adapted – sampling below the Nyquist rate compromises fidelity while higher than Nyquist rate sampling causes inefficiency. Existing approaches to adapt sampling rate tend to err more on the inefficiency side (e.g., by sampling at two different frequencies that effectively doubles the measurement overhead [186]). There are also domain specific approaches to adapt sampling rate (e.g., [116, 117, 126, 132]) but make assumptions on data sparsity, distribution and signal shape that do not hold in general across different network telemetry settings.

Sketching

Sketching is a streaming data summarization method that is aimed at efficient data representation [35]. In the network telemetry context, sketching has been extensively applied for flow-level monitoring tasks [70, 95, 100, 101, 182]. Specifically, this means using a compact data structure (called a sketch) that is essentially a set of counters to represent flow-level information. Sketches are constructed by hash functions to map input flow data to counters, and therefore the information of interest (*e.g.*, flow size) is recorded in those counters with a compact format. With sketching, rather than reporting samples of raw measurements, sketches (compact format of measurement) are reported to the collector periodically. Sketches can also be designed to reserve specific measurements about networks, hence the usage of sketches contributes to greater efficiency for certain telemetry tasks such as heavy hitter detection. Decoding process for received sketches at the collector is also quite lightweight so can be performed in real-time [101].

However, sketching is limited by its very nature trades off versatility for efficiency, as the sketching could be limited to specific system or parameter. The nature of queries to be answered from the data stream must be decided in advance [35]. Also, sketches designed for flow monitoring cannot be used for other types of KPIs (*e.g.*, SNR in cellular networks). The above constraint also affects the overall fidelity with the sketching approach, while it can provide bounded errors to answer queries it is designed for under the assumed conditions, fidelity suffers when those conditions change (*e.g.*, reduction in available memory). For example, more than 30% of the flows can have significant measurement errors with sketching with the memory availability on commodity network elements [69].

Sensing & Recovery

Compressive sensing (CS) [15, 53] is a representative class of methods that fall under this approach. It essentially involves two procedures – sensing and recovery, one at each end. At the sender side, the sensing procedure compresses the measurement data stream (vector) by multiplying it with a sensing matrix. On the receiver side, recovery procedure aims to recover the original measurement vector by solving an optimization problem. While CS has a broad range of applications within and beyond networking (*e.g.*, [23, 33, 90, 108, 196]), its use for network telemetry has been mainly to complement sketching for flow monitoring [69, 71]. For that setting, tailored use of CS has been shown to result in near-zero-errors for almost all flows by ensuring the orthonormality of the sensing matrix [71]. Because of the need to solve an optimization problem at the collector for data recovery, it is difficult for the recovery to keep pace with the telemetry data stream arrival rate and so it faces difficulty in supporting real-time control tasks. Moreover, data sparsity assumption needs to hold for CS to be effective but this may not be true for all network telemetry settings and KPIs of interest.

Another class of methods that take the sensing & recovery approach broadly rely on time-series data compression [18, 34, 46, 191]. These methods are mainly developed for the IoT setting (*e.g.*, to enable efficient data uploading from the IoT gateways to the cloud [34]). The time series data compression or dimensionality reduction done by these methods is relatively more computationally intensive on the sender side and

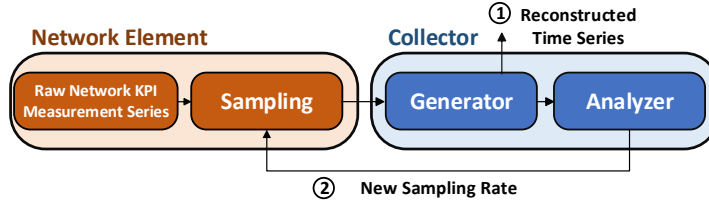


Figure 5.1: Workflow of DISTILGAN. Two stages are: ① data reconstruction and ② sampling rate adaptation.

crucially requires operation over long time windows. This not only limits their versatility due to high processing and memory requirements at the network elements but also makes them unsuitable for telemetry driven real-time network control. Furthermore, we demonstrate through our evaluations that these methods also result in poor efficiency.

Other non-telemetry approaches

Also, relevant are deep learning based data stream recovery and generation methods [49, 52, 91, 106, 112, 162, 188]. Although these methods are not designed for network telemetry, they can be applied for that purpose by pairing them with use of sampling on the sender (network element) side. For a given down-sampled telemetry data stream from the network element, its reconstruction at the collector can be viewed as an ‘imputation’ problem that seeks to recover missing samples. Early approaches in this category (e.g., [91]) focus on recovering a high-quality audio signal from a low-quality one through an interpolation (super-resolution) process using deep convolutional neural networks. More recent approaches are based on generative or diffusion models [52, 106, 112, 162]. As we demonstrate through our evaluations, these imputation approaches are ineffective at low sampling rates (or equivalently, high percentage of missing samples), and also suffer from artefacts when sewing reconstructed data across time windows due to lack of temporal modeling.

The receiver side data stream reconstruction problem can alternatively be seen as conditional sequence generation, i.e., reconstructing/generating original data stream from the down-sampled version received from the sender. Here existing approaches based on deep generative modeling (e.g., [49, 188]) either focus on time domain or on frequency/spectral domain reconstruction, and as a result fail to faithfully recover the features from the other domain. We consider the above set of deep learning methods in our evaluations and demonstrate that from being not tailored for network telemetry, they are inefficient, require significant computing power on the sender side and so not versatile, and have unacceptable inference latency for real-time network control.

The foregoing discussion motivates us to pursue a new and customized approach to network telemetry that is better suited to meeting all the requirements. We present our DISTILGAN approach in the next section.

5.4 DISTILGAN

As stated before, we aim at a network telemetry method that satisfies *high-fidelity*, *efficiency*, *versatility* and *real-time* requirements.

At a high level, our proposed DISTILGAN approach to achieving this aim is through the design of a tailored deep generative model architecture that allows high-fidelity data stream reconstruction at the collector from received sampled data, while simultaneously reducing the transmitted measurement data from the network element by adjusting the sampling rate it in an adaptive manner. Figure 5.1 illustrates our approach. Starting with an initial (bootstrapping) sampling rate, our proposed solution consists of two inter-dependent stages at the collector: (I) Reconstruct data at original granularity from the received sampled data stream through a ‘Generator’ (Figure 5.1: ①). (II) Infer the minimal sampling rate¹ based on the generated data stream via an ‘Analyzer’ and feed it back to the network element to switch to (Figure 5.1: ②).

5.4.1 Overview

Realizing our approach for high-fidelity data stream reconstruction from received sampled telemetry data stream and continually adapting the sampling rate is a significant challenge. A number of targets need to be met to address this challenge:

1. Reconstructing the data stream with high fidelity.
2. Adapting the sampling rate while not compromising fidelity.
3. Achieving general applicability across diverse network telemetry settings.
4. Minimizing inference (data stream reconstruction) latency to support real-time network monitoring and control.

Formally, the first two targets can be stated as follows. Given a discretely sampled form ($\tilde{x}_r$) of a ground-truth telemetry KPI time series $\mathbf{x}(t)$ with raw measurement frequency $r_0 > r$, target (1) translates to generating $x'(t)$ with frequency r_0 so that $x'(t)$ mimics $\mathbf{x}(t)$. As the generation process occurs over short time windows of size W , the input to the generator $\tilde{x}_r = D(\mathbf{x}, r) = \{\mathbf{x}(\frac{d}{r})\}$, $d = 0, 1, \dots, \lfloor \frac{W}{r} \rfloor$. Function $D(\mathbf{x}, r)$ essentially samples $\mathbf{x}$ with sampling rate r . Target (2) mandates that r is kept minimal at any given time instant t . Communication efficiency or compression ratio (CR) achieved through minimal sampling can then be represented by “ $n = 1/r$ ” – higher the n better is the efficiency. For example, if the sampling rate r is $\frac{1}{2}$ of the raw measurement frequency r_0 at the network element, then we achieve $2\times$ efficiency in communication.

To achieve the aforementioned targets, we propose DISTILGAN, a method inspired by temporal super-resolution using deep neural networks. DISTILGAN reconstructs original data stream from received sampled data stream through a tailored conditional deep generative model that operates across both spectrum and temporal

¹For the simplicity of exposition, we focus our description on the sampling rate adaptation but our method also supports adapting the threshold for event/change (e.g., bursts) detection in telemetry data streams, as elaborated in §5.4.3.

domains (Fig. 3.8). It additionally explores available sampling rates automatically, and adapts the sampling rate by reporting the inferred minimal sampling rate to the telemetry data sender (network element).

We achieve target (1) by training a spectral-temporal generator that can produce a fine-grained representation of coarse-grained input sampled data stream, as detailed in §5.4.2. In contrast to a solution that generates a deterministic output, our method learns the inherent stochasticity in the signal and reflects that in the output. To better reconstruct the signal, DISTILGAN incorporates a novel way to fuse spectral and temporal outputs. We further propose a batch generation mechanism to support high fidelity data stream reconstruction on the fly.

As elaborated in §5.4.3, we achieve the adaptive sampling rate target (2) by leveraging the inherent characteristics of our generative model and the outputs it produces with inputs differing in their resolutions. Specifically, we devise a new metric called ‘Q-value’ to quantify the mutual similarity of different rate reconstructions and estimate the data stream reconstruction quality with the current sampling rate. We then decide to increase or decrease the sampling rate guided by the Q-value.

We meet target (3) by relying only on sampling at the network elements, and not making any assumptions on data characteristics (unlike alternative approaches like compressive sensing that assume data sparsity). Sampling is by far the most lightweight approach in terms of processing and memory resource requirements to use at the network elements, and so limiting to sampling at telemetry data senders ensures broad applicability, including over commodity network switches and resource constrained devices. The only additional capability we need from a sender (with negligible processing and memory overhead) is for it to be able to change the sampling rate when informed by the collector (see Fig. 5.1).

Target (4) is achieved through a neural network architecture design that minimally uses computationally heavy RNNs along with additional system optimizations to keep the model inference latency low. We elaborate further on our solutions to meet targets (3) and (4) in §5.4.6.

5.4.2 Generator Model Design for High Fidelity Data Stream Reconstruction

Fig. 5.2 shows the schematic of our generator model architecture. For simplicity, we show the case of reconstructing data stream for a single telemetry KPI from its sampled version. The same approach is applicable for the case to simultaneously reconstruct data streams for multiple different KPIs.

Initial Interpolation

$G_\theta^g : \mathbb{R}^{(L \cdot r)} \rightarrow \mathbb{R}^L$, where L is the data stream length at its original resolution and r is the current sampling rate. Upon receiving the sampled data stream from the sender in a given time window, we first perform the imputation, either with nearest interpolation or IDFT depending on the current sampling rate, so that the data stream has the same resolution as the raw measurement granularity. We further add input-dependent Gaussian noise as latent information to support the stochastic variation from the downstream generative model.

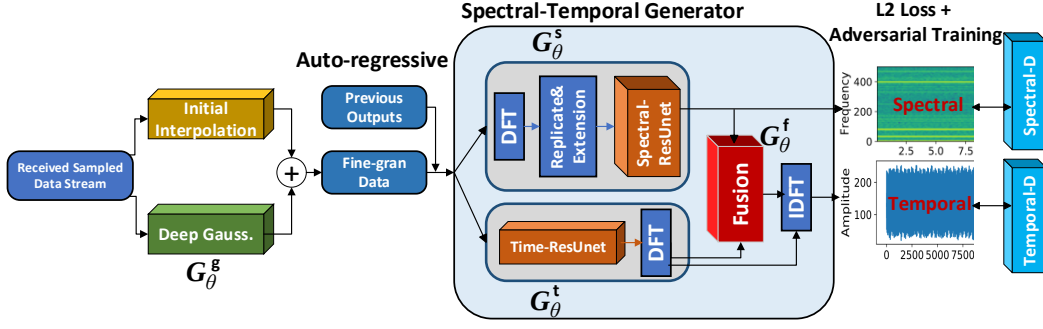


Figure 5.2: Schematic of the DISTILGAN Generator Architecture.

The initial interpolation method is chosen based on the variation range $\Delta = \max(\tilde{x}_r) - \min(\tilde{x}_r)$ as the two paths in Figure 3.8 indicate. More specifically, we have

$$[I_0, I_1] = \begin{cases} [1, 0], & \frac{\Delta_{\text{IDFT}}}{\Delta_{\text{nearest}}} \geq \eta \\ [0, 1], & \frac{\Delta_{\text{IDFT}}}{\Delta_{\text{nearest}}} < \eta \end{cases} \quad (5.1)$$

where the $0 < \eta \leq 1$ is a threshold to select interpolation method, Δ is the variation range after nearest interpolation and IDFT. The intuition behind is that IDFT can reserve low frequency details and nearest interpolation can keep a more accurate dynamic range and they work better in different conditions.

To provide noise input that represents model stochasticity and unobserved context attributes, the interpolated time series is added with a conditional Gaussian noise G_θ^g with learned parameters following [85] as Figure 5.4a illustrates: $\hat{x} = F_{\text{interp}}(\tilde{x}_r) + z_g = F_{\text{interp}}(\tilde{x}_r) + G_\theta^g(x'_{t-W:t-1}, \tilde{x}_r, z_0)$, where F_{interp} is the Nearest Interpolation or IDFT results based on Equation 5.1, z_0 is a noise input following standard Gaussian distribution and $x'_{t-W:t-1}$ is the output of last time window, for which we just use a random initial state sampled from a standard Gaussian distribution for the first time window. Specially for the time stamp with real samples, we keep the original value without adding any noise, and for the other part we add a truncated Gaussian distribution by constraining the noise range with $\delta = 20\%$ (default in this paper) of the sample dynamics ($\max(\tilde{x}_r) - \min(\tilde{x}_r)$) to avoid extreme outlier samples of the learned Gaussian distribution.

Temporal Domain

The temporal domain generator $G_\theta^t : \mathbb{R}^L \rightarrow \mathbb{R}^L$. We design the temporal generator network we refer to as “Time-ResUnet” that is based on the Unet architecture with residual connections [91] but customized to our setting. At a high level, our Time-ResUnet neural network architecture uses downsampling (D-Blocks) and upsampling (U-Blocks) blocks to reduce and increase series length, respectively. The network includes fully connected B-blocks as bottleneck layers. It also consists of Gated Recurrent Unit (GRU) layers at the upsampling end to capture temporal characteristics, with both original input and transformed time series available for output alignment with real samples. Residual connections are used for full utilization of input information.

Figure 5.3a shows a schematic of our Time-ResUnet architecture design. The “D-Block” in Figure 5.3a means the downsampling block that reduces the series length

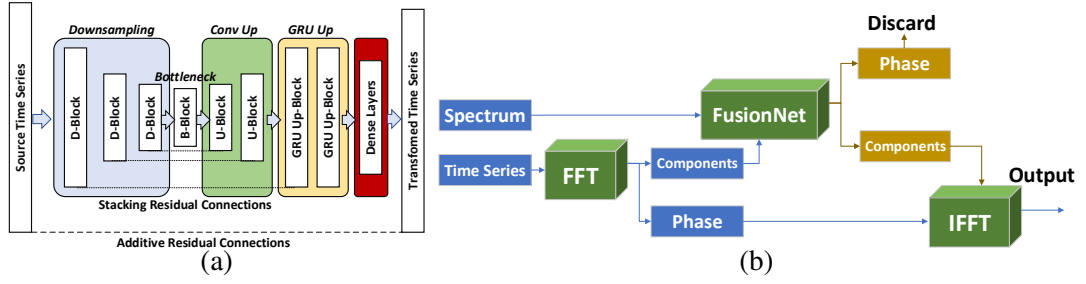


Figure 5.3: (a) Schematic of Time-ResUnet architecture. **D-Block**: 1-D convolutional layers reduces the series length; **U-Block**: 1-D pixel shuffle layers increases series length; **B-Block**: Full connection layers that do change series length. (b) Spectral and Temporal domain fusion. **FusionNet**: 1-D Convolutional Layers.

via 1-D convolution. The “U-Block” means upsampling block, which increases the length of series via a 1-D pixel shuffle method [91, 155]. The residual connections help full utilization of input information across different steps. In the middle there are a few full connection layers called “B-block”, which do not change the length and dimension of the series. The following Gated Recurrent Unit (GRU) layers are applied to capture temporal characteristics. Both the original input and transformed time series are available for GRU layers, and this helps the model to make sure the output is tightly aligned with the real samples.

Spectral Domain

The spectral domain generator $G_\theta^s : \mathbb{R}^L \rightarrow \mathbb{C}^L$. We additionally use a spectral domain generator for enhanced fidelity and better capturing high frequency features but also implicitly for increased efficiency to allow high-fidelity operation with smaller sampling rates. For this generator, after applying DFT to obtain the spectrum as in [49], we keep the spectrum within the current sampling rate and replicate it until the target bandwidth. In this process, the zero frequency component, i.e., power component is kept unchanged to make sure the overall power does not change. Our Spectral-ResUnet neural network architecture design for the spectral domain generation is a variant of the Time-ResUnet architecture in Fig. 5.3a with the only differences being the lack of GRU layers, and operation on complex numbers.

Spectral-Temporal Fusion

$G_\theta^f : (\mathbb{R}^L, \mathbb{C}^L) \rightarrow \mathbb{R}^L$. Through the temporal and spectral domain generators, we get two representations of output time series. While carrying out fusion of these two, we find that the spectral output generally comes with a larger phase noise, whereas the temporal output aligns better with real samples. So if we directly add these outputs, due to the fact that their phase does not match, the model will only learn to reduce the high frequency components to reduce the phase noise, which results in high frequency components getting filtered out in the final output. In order to capture the high frequency components while also reducing the phase noise, we carry out fusion in the spectral domain and discard the phase information afterwards, as shown in Fig. 5.3b.

The overall generator G_θ can be represented as:

$$x' = G_\theta(\tilde{x}_r) = \text{IDFT}\{|G_\theta^f(|\text{DFT}(G_\theta^t(\hat{x})|, |G_\theta^s(\hat{x})|)| \times \text{Angle}(\text{DFT}(G_\theta^t(\hat{x})))\} \quad (5.2)$$

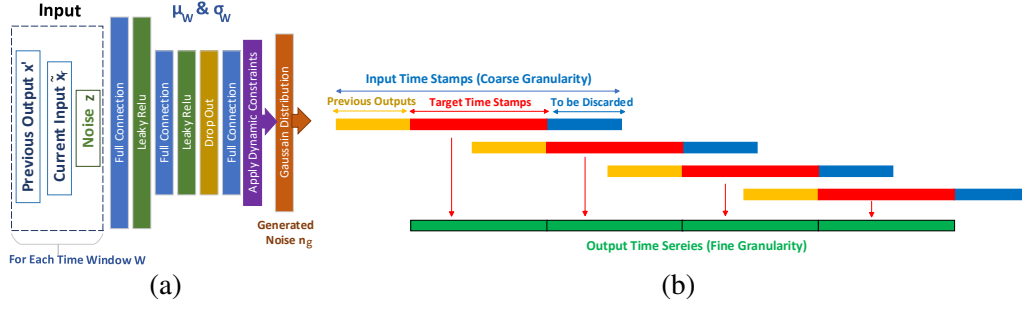


Figure 5.4: (a) Schematic of Generative Gaussian Noise; (b) Batch Generation for Long Time Series

where the function “Angle” in Equation (5.2) represents extracting the angle of a complex vector in the form of an unit complex number, and $\hat{x}$ is the input to the Spectral-Temporal Generator (see Fig. 5.2).

Online Operation with Batch Generation

Online operation here refers to recovering the time series on the fly at every small time window, instead of waiting to receive the whole data stream. Such an operation has three benefits:

- Fast reconstruction without waiting for the whole series
- Enhancing the performance of adaptive sampling by window-wise analysis
- Avoiding artifacts associated with long series generation

We generate arbitrarily long series in small overlapping “batches”; similar approach has been previously shown to be effective for network traffic generation [98]. We further adopt an autoregressive structure: the previous output $x'_{t-W:t-1}$ is included in the input to G_θ^t and G_θ^s . There could be nontrivial artifact when connecting batches if we simply generate them independently because of the different stochastic variations in different batches. To avoid this sewing artifact, we adapt the overlapping window approach in Figure 5.4. For each received window, we discard the last few time stamps of the output and enclose these discarded time stamp in next batch. Meanwhile, we attach the last few time stamps of previous output ahead the current input, making the generation auto-regressive.

Training

To train DISTILGAN, we follow the standard training procedure for conditional GAN [73, 115] using a combination of L2 loss and adversarial training with two separate discriminators respectively for spectral and temporal domain generators. Our spectral discriminator R_s is a multilayer perceptron (MLP) supporting complex numbers, while the temporal discriminator R_t is implemented with single layer LSTM structure and few dense layers.

We train the model by minimizing Jensen-Shannon divergence between the data and model conditionals, *i.e.*, $\theta^* = \arg \min_\theta \text{JS}[p_{\mathcal{D}} || p_\theta]$ following the conditional GAN formulations [73, 115] with the aid of discriminator as in the GAN framework. We denote such discriminator as R due to their role as density ratio estimators [172]. In

our case, we consider two discriminators, named as R_t for temporal thread and R_s for spectral thread. Suppose the spectrum of x and x' are s and s' . Specifically, for given training input measurement data time series, the corresponding adversarial loss between the data $p_{\mathcal{D}}(x)$ distribution and the model $p_{\theta}(x')$ distribution is defined as:

$$\mathcal{L}_{\text{JS}}^{R_t}(p_{\mathcal{D}}, p_{\theta}) = E_{p_{\mathcal{D}}}[\log R_t(x)] + E_{p_{\theta}}[\log(1 - R_t(x'))].$$

$$\mathcal{L}_{\text{JS}}^{R_s}(p_{\mathcal{D}}, p_{\theta}) = E_{p_{\mathcal{D}}}[\log R_s(s)] + E_{p_{\theta}}[\log(1 - R_s(s'))].$$

In our work, the temporal discriminator R_t is implemented with single layer LSTM structure and few dense layers, while the spectral discriminator R_s is a MLP (multi-layer perceptron) supports complex numbers.

We additionally use the standard mean squared error loss for both of the temporal and spectral output:

$$\begin{aligned} \mathcal{L}^M &= \mathcal{L}^{M_t}(x, x') + \mathcal{L}^{M_s}(s, s') \\ \mathcal{L}^{M_t}(x, x') &= \frac{1}{L} \sum_{t=1}^L (x - x')^2, \quad \mathcal{L}^{M_s}(s, s') = \frac{1}{L} \sum_{t=1}^L (|s| - |s'|)^2 \end{aligned}$$

where s and s' is ground truth spectrum and generated spectrum (as complex number). Overall, together with the adversarial (GAN) loss, the loss function to fit θ is:

$$\mathcal{L} = \mathcal{L}^M + \lambda_t \mathcal{L}_{\text{JS}}^{R_t} + \lambda_s \mathcal{L}_{\text{JS}}^{R_s}$$

where the λ_t and λ_s is a weight to balance the effect of adversarial loss, which in our case is set as $\lambda_t = 0.1$ and $\lambda_s = 0.1$ by default.

The learned Gaussian noise is truncated by $\delta = 20\%$ of the absolute range of raw input samples of corresponding time window. A large δ leads a stronger stochastic variation on the output but DISTILGAN can easily learn how to denoise the outlier samples if the input is not very bursty. In the Time-ResUnet, we only use 1 GRU layer because more GRU layers only have marginal enhancement on the performance with extra increase in the inference time.

We use single layer LSTM network with dense layers (three full connection dense layers) in the DISTILGAN temporal discriminator, and complex MLP network for spectral discriminator.

We use $W = 96$ for the reconstruction time window size by default and the default step length during training is set to 15, so there is overlap between training batches. $W = 96$ is sufficient for the dataset in this paper— larger W does not show significant gain, whereas too small W makes high CR experiment hard to carry out. Note that, in our experiments, we found that any step length as defined in §5.4.2 between 1 and 15 gives similar result.

5.4.3 Efficiency through Sampling Rate Adaptation

Here we first present how we estimate the received data stream reconstruction quality and then describe how we use it to adapt sampling rates.

Reconstruction Quality Estimation for Adaptivity

Data driven methods can aid reconstruction quality estimation. In machine learning, model uncertainty estimation has been a subject of investigation in recent years. Methods from that body of literature [58, 152, 165] can shed light on understanding the possible error during the model inference. However, these existing techniques come with high computational overheads and so unsuitable for our purpose:

- *Monte Carlo Dropout* [58] – This method requires repeated inference to estimate the learned distribution, which conflicts with our real-time inference requirement. It also uses a dropout layer for which the dropout ratio needs to be fine-tuned.
- *Conformal Prediction* [152, 165] – This method requires training an error prediction on different quantiles, which is computationally heavy contrary to our real-time requirement. The benefit from this method is also limited when the underlying variation in the data is large.

In light of the above, we seek to devise a lightweight method to estimate the data stream reconstruction quality and use it in turn to guide our sampling rate adaptation. We observe that even without access to fine-grained ground truth time series for the KPI of interest, its reconstruction results could still be used to estimate the recovery quality together with the model. Our key insight is that: (1) there exists a sampling rate at any given time which is sufficient to ensure high quality reconstruction; and (2) the model would give similar quality result even if the sampling rate is further increased beyond that point of sufficient sampling rate.

Specifically, we use an intuitive measure of the recovery quality based on the robustness of recovery under different levels of sampling redundancy, where the robustness refers to the extent to which the model can tolerate more and more missing samples. In other words, we would like to “ask” the model about its estimated reconstruction quality on the result. In particular, we define the estimated quality metric, *Q-value*, as follows:

$$Q(G_\theta) = \sum_{i=1}^{N_u} \text{NMAE}(G_\theta(D(x, r)), G_\theta(\underline{D}(x, r - i))) / N_u, \quad (5.3)$$

where N_u is a configurable parameter² representing the other lower sampling rates immediately below the current sampling rate to be examined; r is the sampling rate used for the current time window;

NMAE is the normalized mean absolute error (§5.5); $D(x, r)$ is the current received sampled data stream; and function $\underline{D}(x, r - i)$ is the variant of D that represents sampling at the lower rate of $r - i$.

In §5.4.4, we provide theoretical justification on the correlation of the above Q-value metric with reconstruction error of the model. In §5.4.5, we further provide supportive empirical evidence of this correlation considering different scenarios.

The above suggests that we can rely on the Q-value to estimate the effect on data stream reconstruction quality from changing the sampling rate. If Q-value is very low

²In our evaluations, we empirically set this parameter $N_u = 2$.

then this indicates that the current sampling rate may be higher than sufficient and so could be adjusted down. Otherwise, sampling rate should be increased. This raises the question on the scale by which increase and decrease of sampling rates should happen. In §5.4.5, we examine additive/multiplicative policies for sampling rate adaption (à la TCP congestion avoidance mechanism).

Adapting Sampling Rate in DISTILGAN

In DISTILGAN, the capability to adapt sampling rate is reflected along two inter-dependent procedures:

- *Automatic Sampling Rate Space Expansion:* DISTILGAN will automatically find applicable sampling rates, as depicted in Fig. 5.5a. This automatic sampling rate expansion starts from finer granularity pre-trained models. When the existing pre-trained models all show similar reconstruction quality performance, DISTILGAN will train an even lower sampling rate model and try to reconstruct the received data stream sampled at that rate. If the quality is maintained as inferred through Q-value, then the new lower sampling rate will be added into the rate space for real-time sampling rate adjustment from the next time window. With this mechanism, the telemetry system does not have to train the model for all possible sampling rates in advance. Instead, DISTILGAN can organically expand the rate space and corresponding trained models over time.
- *Automatic Sampling Rate Selection:* DISTILGAN will select from available sampling rates in an adaptive manner, as illustrated in Fig. 5.5b. The selection procedure is as follows:
 1. the received sampled data stream is down-sampled with sampling rates lower than the current one to create further sampled data streams. The reconstructed data streams corresponding to those further sampled streams (i.e., at lower rates) are generated by the model.
 2. If Q-value scores from above step (1) are lower than a small threshold, then the lowest sampling rate that does not degrade the reconstruction quality is selected.
 3. Otherwise, the current sampling rate is increased and set as the selected rate for use by the sender in the upcoming time window.

If after the above procedure, the current rate matches with the selected rate then rate space expansion is triggered to train models at sampling rates lower than the current rate, following the procedure outlined in Fig. 5.5a.

Threshold based Sampling

DISTILGAN uses threshold based sampling to capture the significant and random burst in time series³. Mathematically, this part captures top $k\%$ of the samples, to avoid

³By default, we provide the same threshold based sampling result as DISTILGAN to all baselines for fair comparison, as correct threshold is unavailable (see §5.6.2) for the model that cannot learn correct distribution. Most of samples still measured by periodical sampling.

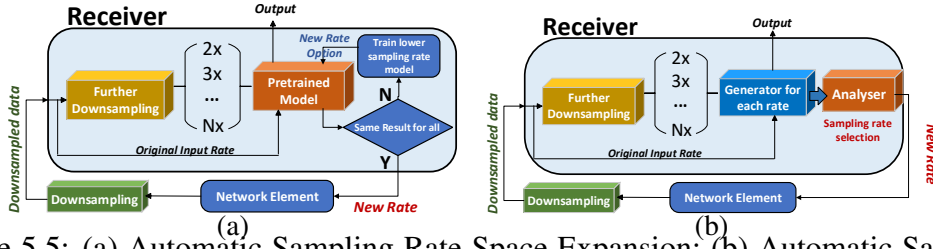


Figure 5.5: (a) Automatic Sampling Rate Space Expansion; (b) Automatic Sampling Rate Selection.

missing extreme burst. The threshold is learned by DISTILGAN, we use the past distribution generated by DISTILGAN to represent the ground truth distribution because the generative model can easily capture the conditional distribution of the data. As for the quantile of the threshold, without prior measurement of data sparsity, we only apply a small quantile for significant outliers, which is in general less than top 5%. We then leave the adaptive part to periodical sampling because it is also able capture enough information when the redundancy is low by increasing sampling rate. Threshold based sampling is especially important to applications like 5G Rand and IoT for which signal burst is important; we detail the usage in these applications in §B.5.2.

5.4.4 Q-value as coarse recovery error estimation

Q-value can be taken as an estimation of recovery error when the following two conditions are met. Suppose we are targeting on a fidelity loss ϵ that is a very small value (e.g., NAME < 0.01)

Condition 1: Further subsampled series should have a decreasing recovery quality:

$$\text{NMAE}(G_\theta(\underline{D}(x, r - m)), x) \leq \text{NMAE}(G_\theta(\underline{D}(x, r - n)), x), \text{ If } m \leq n \quad (5.4)$$

Condition 2: The adaptive step is large enough to bring significant change to the output when there is no redundancy in the input data. There is a Δ leads to:

$$\text{NMAE}(G_\theta(\underline{D}(x, r)), G_\theta(\underline{D}(x, r - \Delta))) \geq \epsilon, \text{ if } \text{NMAE}(G_\theta(\underline{D}(x, r)), x) > \epsilon \quad (5.5)$$

If the current sampling rate has significant redundancy, and even with lower n sampling rates ($n \geq 1$) the model can still reconstruct the original series, then the first n components in the summation of Q-value (Equation 5.3) should be very close to zero. Otherwise, if there is no redundancy at all, the further subsampled time series should lead to a distinct output, and further reducing sampling rate can only make the difference on reconstruction more significant (Condition 2), until the performance is close to unconditional generation — higher reconstruction loss than any sampling rate, when the sampling rate is extremely low. When i is small:

$$\text{If Redundancy : } \text{NMAE}(G_\theta(\underline{D}(x, r)), x) << \epsilon, \text{ and } \text{NMAE}(G_\theta(\underline{D}(x, r - i)), x) \leq \epsilon \quad (5.6)$$

$$\text{Then : } \text{NMAE}(G_\theta(\underline{D}(x, r)), G_\theta(\underline{D}(x, r - i))) \leq \epsilon \quad (5.7)$$

If a low sampling rate is insufficient or current sampling rate is significant lower than required, all re-sampled result should have different result and thus the Q-value is

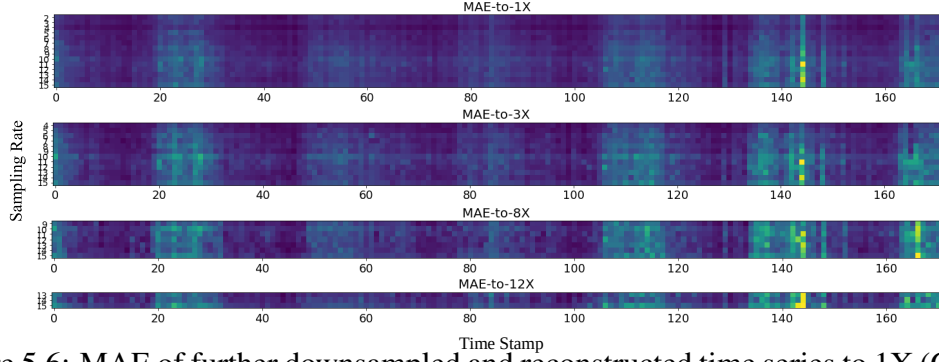


Figure 5.6: MAE of further downsampled and reconstructed time series to 1X (Ground Truth), 3X, 8X, and 12X (reconstructed) time series in IoT Smart Metering

larger, if **Condition 1&2** is true.

$$\text{If No Redundancy : } \text{NMAE}(G_\theta(D(x, r)), x) > \epsilon, \quad (5.8)$$

$$\text{Then : } \text{NMAE}(G_\theta(\underline{D}(x, r - \Delta)), x) > \epsilon, \quad (5.9)$$

$$\text{and } \text{NMAE}(G_\theta(D(x, r)), G_\theta(\underline{D}(x, r - \Delta))) > \epsilon \quad (5.10)$$

Now discuss how to make **Condition 1 and 2** be true at the same time.

- **Condition 1** is in general work in our dataset, which can be verified by experiments. The less input information, the worse recovery quality it is.
- **Condition 2** is true when the Δ is large enough to introduce significant change in the input. Suppose $\Delta = r$, the model will work on unconditional state, the difference to conditional generated result should be significant and larger than the target fidelity loss ϵ .

Both of the condition can be directly verified through experiments. This phenomenon exists in all the three datasets in §5.5, and therefore the generality should be promising.

As stated before, in this paper we use a $N_u = 2$, so that the maximum value of i is 2. Although we use a small N_u , the Q-value is sufficient to detect micro changes of the input — if the sampling rate is higher than loss-less recovery rate, then Q-value is around zero, otherwise the model need to increase the sampling rate to make sure there are certain redundancy in the samples. We also tried larger N_u in ISP network, the result is showed in Figure 5.7b: it is too sensitive by giving an error estimation much larger than actual value, which might cause memory wasting.

We further evaluate the effectiveness of Q-value in §5.4.5, and the results show that the Q-value is sufficient to adjust actual sampling to the optimal rate — minimal sampling rate that meets fidelity requirement.

5.4.5 Mutual Difference of Reconstructed Time Series

Mutual difference of reconstructed time series refers to the components in the summation of Q-value and inequality (5.5). We verify **Condition 1** and **Condition 2** in §5.4.4 directly by experiments. Taking IoT Smart Metering for example, from Figure 5.6 we have observed that when the reconstructed time series exhibit a higher NMAE to the

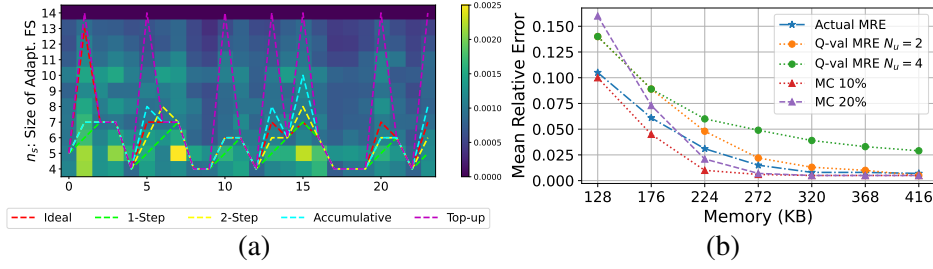


Figure 5.7: (a) Comparison between different AIMD strategy on CAIDA dataset (flow size by $\langle \text{source}, \text{destination} \rangle$), at 1 second epoch length. Background: Normalized MAE (NMAE); (b) Actual Error, Q-value, and Monte-Carlo Estimation with different FS [71] size in DISTILGAN based sketching.

ground truth compared with other time stamps, the reconstructed time series will also show a higher mutual difference. This verifies the high-level effectiveness of the Q-value. From Figure 5.6 we also observe that lower sampling rate tends to have worse reconstruction fidelity and larger mutual difference, and therefore the Condition 1 is verified. We also provide another example on ISP network Figure 5.7a illustrates different sampling rates for different runs of AIMD algorithms in §5.4.3 on ISP network traffic. Overall the phenomenon matches the example in IoT smart metering—further downsampling contributes to worse mutual NMAE, and the AIMD algorithms do help to adapt to a suitable sampling rate. For Condition 2 we show the result in Figure 5.7b, where we prove that if the N_u is configured with proper value, the Q-value will guide the model to converge to expected loss. In Figure 5.7b, the memory represents the input information in the sketching system, similar to the sampling rate in other cases. When the memory is much lower than necessary, the Q-value is even higher than the actual loss. As the memory increases, the Q-value converges to a small value simultaneously with the actual loss. Larger N_u means higher sensitivity as the $N_u = 4$ case in Figure 5.7b because larger i will be considered when compute Q-value, and then it needs more redundancy on sampling rate to meet the target Q-value.

5.4.6 Design Choices & Optimizations for Versatility and Real-Time Inference

In addition to the aspects highlighted earlier in §5.4.1, the versatility and real-time nature of DISTILGAN is reflected through the following aspects:

- *Minimal processing and memory requirements at the sender.* As stated before, sampling is a lightweight process at the network elements. All that the sender needs to do with DISTILGAN is to simply report measurement KPIs of interest to the collector at the specified sampling rate. The above combined with the adaptively use the minimal sampling rate throughout keeps the resource requirements at the network elements at the bare minimum that even the most resource constrained elements can cope with.
- *Low inference latency:* millisecond level inference for recovering fine-grained time series of measurement KPIs of interest.
- *Fast Training:* Ability to achieve good generalization with small amount of training data.

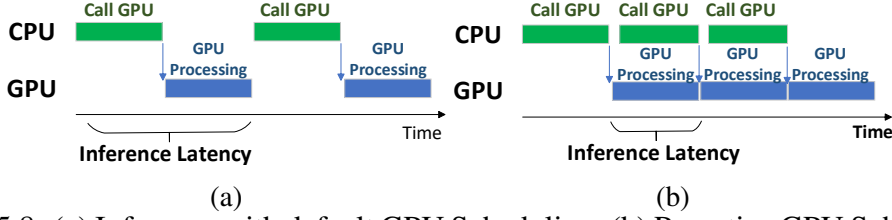


Figure 5.8: (a) Inference with default GPU Scheduling; (b) Proactive GPU Scheduling.

	GPU (w/o Parallel)	GPU (in Parallel)	CPU (w/o Parallel)	CPU (in Parallel)
Avg. Inference Time (ms)	1.8	1.2	17.8	12.5

Table 5.2: Inference Latency with DISTILGAN.

For network switches, the benefit of DISTILGAN would be mainly in bandwidth savings and efficiently using network resources with minimal monitoring overhead. However, when performing telemetry with resource constrained and battery operated mobile and IoT devices, minimal sampling rate (and therefore longer intervals between reporting samples) can be directly converted into significant power savings. As an example, with discontinuous reception (DRX) mechanism in 5G, reducing the measurement frequency from 100 Hz to 25 Hz ($4\times$ reduction in sampling rate) results in up to 80% power saving on radio module and around 40% power saving on commercial smartphones, according to the measurements result of major operators [7].

We measured and optimized the execution of DISTILGAN. First, the spectral and temporal generator processes can be executed in parallel. Second, since the samples arrive every time window, we can further reduce the CPU I/O and scheduling time through proactive scheduling, as illustrated in Fig. 5.8. We also try executing DISTILGAN solely over CPUs. The experiments are based on NVIDIA Tesla A100 Ampere 40 GB GPU and AMD EPYC 7302 16-Core CPU. Results are shown in Table 5.2 using received sampled stream with a 96-sample window as input to DISTILGAN generator. Leveraging GPUs and parallel execution via proactive scheduling, DISTILGAN can achieve a $1.2ms$ inference latency. Even the $12.5ms$ CPU execution latency is clearly sufficient for the IoT smart metering and 5G RAN monitoring scenarios with 1s and 1ms measurement granularities, respectively (see Table 5.3). For example, with 1ms granularity and window size of 96, $12.5ms$ CPU execution based inference latency will ensure data recovery 83.5ms earlier than the arrival of next window of samples. Even when considering the 100 microseconds flow level measurement granularity in general ISP networks, $1.2ms$ GPU execution latency is well within the 10ms time budget to process a 96-sample input window. In fact, the above inference latency results suggest the ability DISTILGAN to comfortably recover larger windows of incoming samples without any need for buffering or waiting times at the collector.

The training of DISTILGAN is fast. For the network scenarios considered in this paper, we can expect training time ranging from tens of seconds to a few minutes with a single GPU, depending on the scenario and window size. At the same time, DISTILGAN achieves a good generalization with a small randomly chosen training set. For ISP networks and 5G RAN, DISTILGAN when trained on less than 10% data of each dataset achieves good performance on the remainder of those datasets, while for the IoT smart metering scenario the amount of training data required is much lower at 3%. If the measurement of ISP network and 5G RAN lasts over a longer period as one would typically expect it to be the case, we can expect a good generalization with

much smaller percentage of training data. We explored such a situation with synthetic datasets (§B.6.1) and found that we only need to train DISTILGAN with less than 1% of whole synthetic time series to achieve good generalization.

DISTILGAN barely has any computational or memory overhead on the network element end due to use of sampling at minimal rates. DISTILGAN also has the potential for profound benefits for telemetry with resource constrained and battery powered network devices, such as significantly reducing the power consumption and extend the battery life cycle when combined with sleep mechanisms. We can reduce the inference latency of DISTILGAN to 1.2 ms with GPUs and 12.5 ms with CPUs, which is sufficient for most real-time network monitoring and control applications. To achieve the aforementioned benefits while ensuring high-fidelity data stream reconstruction, DISTILGAN only requires a small amount of training data for a good generalization, generally less than 10% of the entire dataset in our experiments.

5.5 Evaluation Methodology

5.5.1 Datasets

We evaluated DISTILGAN using three real-world scenarios (Table 5.4) with associated datasets (Table 5.3). We also evaluated DISTILGAN on synthetic datasets (sine wave and fGn [17]) in §B.6.1 to represent some typical signals in engineering. By default, we use first 20% time series as training data, and evaluated the model on the rest of the 80% data.

ISP Network. The performance in ISP network is evaluated on Trace-driven simulations with CAIDA Traces [25], which is a representative real world ISP network traces at large scale with $100K \sim 150K$ TCP flows every second. This dataset has been used in various works in network telemetry [69, 71, 199], and we use the same data source to make sure the evaluation is done with the target scenario of certain baselines.

5G RAN. We leverage the state-of-the-art JANUS RAN monitoring system [55] in a carrier-grade 5G Open RAN (O-RAN) testbed and investigate the feasibility of minimally sampling representative radio network KPIs and recovering the fine-grained radio KPI time series afterwards. This O-RAN testbed allows measuring a large set of KPIs but some of them like Cell ID, Number of UEs, number of resource blocks (RBs), etc., do not vary with fine granularity, so we do not include them in our experiment. Based on our investigation on all radio KPIs that are measured frequently in O-RAN, we select three key KPIs that are reported with fine granularity and has significant variation in small time scales: Signal to Noise Ratio (SNR) – signal quality; Downlink Data Transport Block Size (DL TBS) – very fine granularity throughput measurement; and Timing Advance (TA) – mobility measurement.

IoT Smart Metering. IoT system Smart Metering [34], is multivariate dataset from four real smart meters installed at different locations in a university campus. This dataset has the longest time scale (3 months) with $1s$ granularity, and also is particularly suitable for adaptive sampling rate evaluation. The set of KPIs included in this IoT dataset is listed in Table 5.4.

	CAIDA	5G RAN	IoT (Smart Metering)
Time Granularity	10ms*	1ms	1s
Variate Number	1	3	6
Flow Number	5×10^5	1	1
Longest Series	1 hour	1 hour	3 months
Total Time Stamp	3.6×10^5	3.6×10^6	2.5×10^5

Table 5.3: Measurement statistics for different datasets. *: 10ms is the minimal granularity we consider to query network flow information such as size.

Dataset	KPIs (Measurement Attributes)
CAIDA ISP Network (1)	Flow size
5G Radio Network (3)	Signal-Noise Ratio (SNR), Downlink Transport Block Size (DL TBS) and Timing Advance (TA).
IoT Smart Metering (6)	Apparent Power, Power, Energy, Current, Frequency, Voltage.

Table 5.4: KPIs considered in different datasets.

5.5.2 Metrics

We evaluate the performance of DISTILGAN in comparison with a wide range of baseline methods representing the state-of-the-art (Table 6.5), focusing on all four essential requirements for network telemetry methods highlighted earlier in the motivation section (§5.2)⁴. In the following, we describe which specific metric we used for each of the requirements.

ISP Network settings — P(Err.;0.1%). Here we focused on the ability to reconstruct flow sizes. For this, we use the percentage of flows that have less than 0.1% relative error as the metric, following the state of the art work on flow-level monitoring in ISP networks [71]. The recovered size is $Size_r$, and original size is $Size_o$, then relative error $Err. = \frac{|Size_o - Size_r|}{Size_o}$.

5G RAN and IoT Smart Metering settings— NMAE, NWD, and NSWD.

Normalized MAE (NMAE). Given a measured data time series indicated with $(x : \{x_1, x_2, \dots, x_T\})$ and the generated one $(y : \{y_1, y_2, \dots, y_T\})$, the Mean Absolute Error (MAE) is calculated with the following: $MAE(x, y) = \sum_{i=1}^T |y_i - x_i| / T$. NMAE is the MAE after normalization:

$$NMAE(x, y) = MAE(normalize(x), normalize(y)),$$

where $normalize(x) = \{\frac{x_i - \min(x)}{\max(x) - \min(x)}\}_{i=1}^T$. Compared to the original MAE, NMAE can better reflect the overall fidelity as it has been also advocated in past works [21, 194].

Normalized Wasserstein Distance (NWD). This metric help us quantifying the similarity between the reconstructed time series and its original counterpart. We compute the Wasserstein Distance (WD) [148] after data normalization to obtain NWD, as also used in [187]. This metric can reflect the accuracy of percentile threshold.

Normalized Spectrum Wasserstein Distance (NSWD). Besides computing the WD on temporal domain, we also measure the spectrum similarity with WD. Same as the temporal correspondance above, we normalize the bandwidth before computing WD; the DFT is computed with a 1000 window size, around 10 times of default

⁴For detailed description of various baselines, please see §B.6.

window size of reconstruction, to obtain an reliable result.

Efficiency. We use the **Compression Ratio (CR)** as a measure of efficiency. Generally speaking, compression ratio c_r is defined as $c_r = Size_o / Size_a$, where $Size_o$ is the memory cost of the raw measurement while $Size_a$ of the one actually stored.

In the case of sampling and sampling based baselines (including sensing & recovery based methods), c_r can be seen as the ratio of the raw measurement frequency to the sampling rate used at any given time instant. With sketching, the size of the sketches and their frequency of reporting determine the c_r .

Versatility. We demonstrate the versatility of DISTILGAN by applying our solution to the three very different telemetry settings presented in Table 5.3. Additionally, we show the behavior of our system under different constraints (e.g., memory, measurement window, number of flows etc.)

Real-time. We used the *data stream recovery time* as the metric for this requirement. As discussed in the previous point, the data is not stored raw in memory. This metric allows to evaluate how much time it takes to recover the original data from the compressed version. If the recovery time is smaller than the epoch length (input window size) used for data generation, then the recovery can be considered as real-time.

5.6 Evaluation

In this section we compare DISTILGAN against state-of-the-art approaches (Table 6.5). We focus on its efficiency (§5.6.1), versatility (§5.6.2) and real-timeliness (§5.6.3) while keep high-fidelity measurements reconstruction as a general requirement for all the experiments. We evaluate our system using the three scenarios presented in Table 5.4: ISP network, 5G RAN, and IoT Smart Metering. In the Appendix (§B.4) we share additional information about specific configurations and tests methodology used throughout the evaluation.

5.6.1 DISTILGAN achieves High Fidelity and Efficiency

We evaluate the efficiency comparing the maximum CR while guarantee a pre-set measurement fidelity. In Figure 5.9a, we compare the behavior of DISTILGAN against state-of-the-art sketching solutions when estimating flow sizes from a real network trace taken from CAIDA. Here we set our measurement time window (*i.e.*, epoch length) to $10ms$ as real-time network telemetry systems need a very small control loop. From the figure, we can see that DISTILGAN allows for up to $25\times$ gain in terms of bandwidth consumption than state-of-the-art when targeting on near-zero error measurements. Classical CS based methods such as SeqSketch [71] and EmbedSketch [71] are not efficient when used with very small time windows because the orthonormality of sensing is not well guaranteed with their design, the flow size distribution on ms level is different to the distribution on seconds level (B.4.1), and hence need to apply more counters, whereas DISTILGAN is robust to lack of orthonormality after training.

We also compared DISTILGAN against solutions performing sensing & recovery using the same ISP network scenario with CAIDA. Considering the operation of these systems, for a fair comparison, we did not use a time window but we sampled the original network trace and tried to reconstruct the missing information. As Figure 5.9b

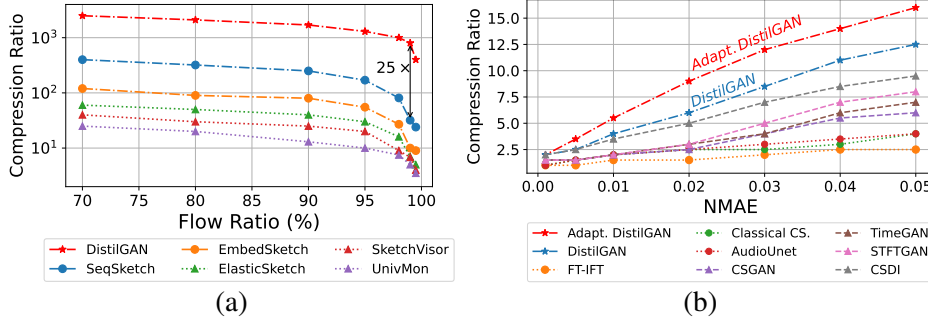


Figure 5.9: Flow-level monitoring in ISP networks. (a) Comparison with sketching methods using compression ratio vs. flow ratio (percentage of flows with less than 0.1% relative error on flow size); (b) Compression ratio vs. fidelity (NMAE) for methods not using sketching – just sampling IP packets.

shows, the feasible compression ratio is lower than the sketching case, as the models need to recover the size of missing packets, instead of just the whole flow. Overall, generative models yield significant gain in terms of efficiency, up to $8\times$ better than conventional data-driven solutions. We also observe that DISTILGAN shows better CR than any of the considered generative models under same fidelity requirement. Besides, compared with constant sampling rate case, the adaptive mechanism bring about 25% $\sim$ 35% gain on CR.

Finally, in Figure 5.10, we demonstrate the high-fidelity nature of DISTILGAN in being able to capture high frequency components using the same baselines as in Fig. 5.9b. For this, we show the spectrum and amplitude from doing a sinusoidal wave signal reconstruction. The synthetic signal has few components around 400Hz, 100Hz, and 50Hz. For visualization, we give the power on 400Hz, 100Hz, and 50Hz with more significant value so the main components are clear. The Nyquist rate is 800 Hz and the signal is sampled at 200 Hz. All the methods work on window size at 6 — *the model is required to output reconstructed signal after receiving every 6 samples*. In Figure 5.10b we can observe that the high frequency components are lost with FT-IFT, while in Figure 5.10c we observe all three components. Without learn spectrum explicitly, CSDI can hardly identify the components below 100Hz but simple generate random noise spectrum beyond that. STFTGAN is the only other method in the baselines that can identify the high frequency features, however, it cannot avoid the artifacts introduced by recovering from a short window and the noise level much higher than DISTILGAN. The phenomenon here is aligned with the NSWd performance in Figure 5.11c and Figure 5.12c.

5.6.2 DISTILGAN is Versatile

DistilGAN seamlessly works across different telemetry domains

Besides ISP network, we also evaluate DISTILGAN on 5G RAN and IoT Smart Metering scenarios, where the main task is to reconstruct the original time series (i.e., stream of measurement data) after temporal downsampling. Such task is common in 5G RAN and IoT system for the various purposes, *e.g.*, saving bandwidth. We carry out experiments using the KPIs presented Table 5.4, and take the average performance

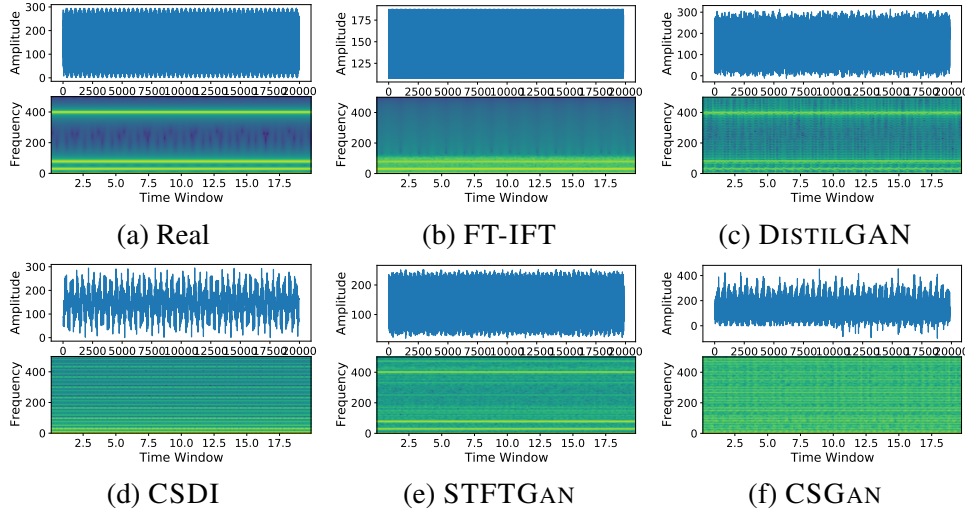


Figure 5.10: Sinusoidal wave reconstruction with different methods

of all KPIs in each scenario as the final performance. In Figure 5.11 and Figure 5.12, we show how NMAE, NWD, and NSWD evolve when changing CR. Here we compare DISTILGAN against state-of-the-art techniques in the sensing and recovery world.

As Figure 5.11 shows, DISTILGAN in general has the best fidelity in terms of all metrics with different CR. Compared with FT-IFT, the adaptive DISTILGAN has 75% better NMAE, and more than one order better than FT-IFT in terms of NWD and NSWD. Compared with other data driven solutions, in Figure 5.11a, the gain of DISTILGAN on NMAE is marginal when the CR is lower than 8, but when CR is higher, DISTILGAN shows much better capability on maintaining the performance on NMAE. As for NWD (Figure 5.11b) and NSWD (Figure 5.11c), DISTILGAN shows almost 1 order gain to the baselines, indicating the superiority of DISTILGAN in distribution learning on both of temporal and spectral domain.

Without an adaptive mechanism, all methods start to have much worse performance if CR goes beyond 8. By tuning the Q-value, the adaptive version of DISTILGAN has more robust performance when CR changes. CSDI is good at NMAE when the CR is lower and has similar performance on NWD with DISTILGAN, but it cannot learn the spectrum efficiently and the NSWD performance is up to 10 times worse than DISTILGAN. Spectrum can be recovered with high-fidelity only with the model that learn the spectrum explicitly, such as DISTILGAN and STFTGAN.

The performance evaluation in IoT smart metering system is showed in Figure 5.12. Here we use the same methodology as in 5G RAN, and overall all data-driven methods show better fidelity in this specific dataset. Such observation is aligned with the fact that our smart metering data comes with much coarser granularity and higher redundancy. Even though, DISTILGAN still advances the other methods by having equally good performance on all three metrics.

DISTILGAN is Robust

We now evaluate the behavior of DISTILGAN when estimating the flow size of network traffic and changing either epoch length (i.e., time window) or system condition.

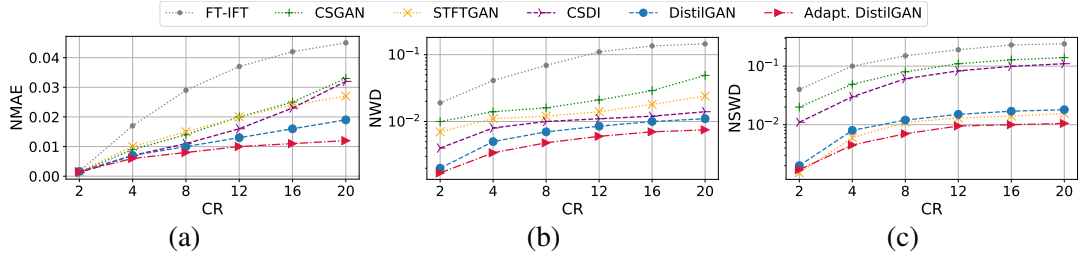


Figure 5.11: Fidelity vs. compression ratio in 5G RAN, in terms of: (a) NMAE, (b) NWD, (c) NSWD.

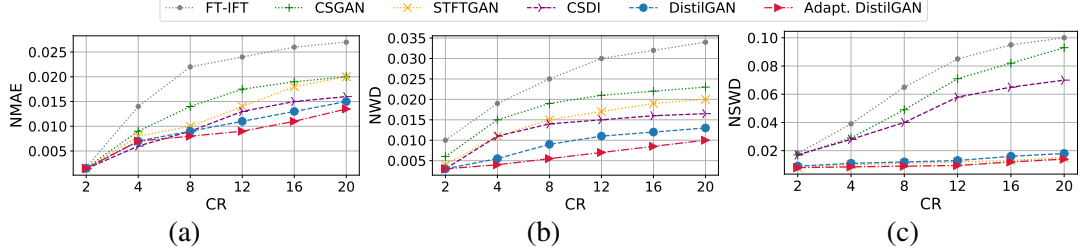


Figure 5.12: Fidelity vs. compression ratio in IoT Smart Metering, in terms of: (a) NMAE, (b) NWD, (c) NSWD.

We compare this against state-of-the-art sketching based approaches that also adopt a time window for their measurements. Starting from CAIDA traces, we varied the epoch length from 10ms to 1s, kept the fidelity constant and evaluated the CR. DISTILGAN has up to $25\times$ gain on bandwidth efficiency when the epoch is very short, and still use less bandwidth than the others even for 1s epoch length (Figure 5.13a).

We then varied the available memory and evaluate the accuracy of flow size estimation. Here we observe that DISTILGAN can provide high-fidelity measurements even with very limited memory: with only 450KB available, it can still get $\sim 80\%$ of measurement correct.

Finally, as Figure 5.13c shows, the gain on memory efficiency brought by DISTILGAN is constant regardless the epoch length.

5.6.3 DISTILGAN is Real-Time

Overall Recovery Time

In this section, we focus on the ability to provide results in real-time (e.g., performing data reconstruction faster than arrival rate of new measurements). We consider a worst case scenario, where a sketch-based solution needs to estimate the flow sizes of 100K flows, while a method based on sampling or sensing and recovery needs to reconstruct 100K KPIs. Based on our datasets, 100K flows/KPIs tend to appear every 1s and hence we consider an approach real-time if it is capable of producing a result in less time.

In Figure 5.14, we compare the recovery time for all the solutions we have considered so far. Methods that leverage CS (i.e., SeqSketch, EmbedSketch, and SketchVisor), together with Classical CS approaches might require $> 60s$ to get the results, so they cannot be considered to be real-time. Deep diffusion based methods like CSDI need hundreds times longer inference time to get a reliable result than simple DNNs, and hence latency is high with CSDI.

Some of conventional sketching methods such as UnivMon and ElasticSketch meet

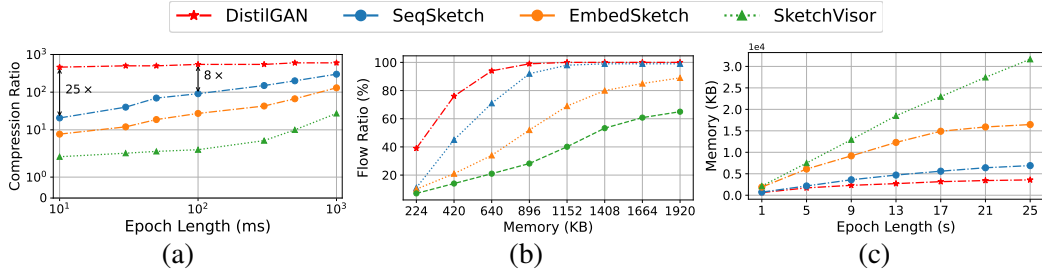


Figure 5.13: (a) Comparing with sketching methods on compression ratio in ISP Network for different epoch lengths; (b) Ratio of flows with less than 0.1% relative error vs. memory; (c) Memory Requirement to meet high-fidelity requirement under different epoch lengths.

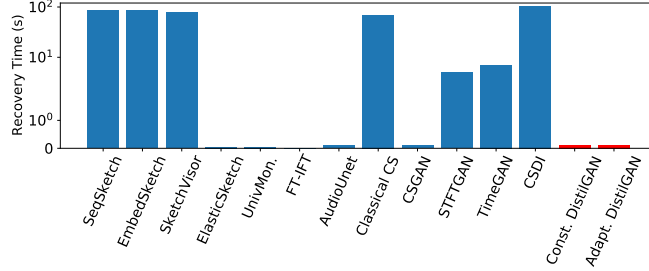


Figure 5.14: Recovery time to get measurement result with 100K flows for Sketching methods: SeqSketch, EmbedSketch, SketchVisor, ElasticSketch, and UnivMon; or 100K KPIs for other methods including DISTILGAN.

the real-time requirement well because they do not need complex computation. Sub-sampling and then Inverse Fourier Transform (FT-IFT) can be done in real-time, but the fidelity is worse than data driven methods in §5.6.2. Some deep learning based methods might be two orders faster in recovery time than classical CS (e.g., AUDIOUNET, CSGAN, and DISTILGAN), but this only happens when the neural network structure is simple (e.g., AUDIOUNET and CSGAN) or fine-tuned for real-time (e.g., DISTILGAN).

Real-time performance for specific task

We run further tests to better understand the results presented in the previous subsection. In Figure 5.15a, we show the inference latency on a single core CPU of DISTILGAN, SeqSketch and SketchVisor when varying the number of flows. Here, we can see that the inference time only increases near linearly with the number of flows, and it never needs more than 1s, even in the presence of 1000K flows to decode. Noteworthy that DISTILGAN still perform up to 90% better, especially its implementation on a GPU.

Then for 5G RAN, we focus on sampling and recovery task. We measure the optimal temporal window size in Figure 5.15b when work on the 5G RAN dataset in Table 5.4, and then measure the inference latency with the optimal temporal window size in Figure 5.15c. From Figure 5.15b, we can see that classical CS and AUDIOUNET requires larger window size to achieve the best CR. Classical CS, CSDI, TIMEGAN, and STFTGAN cannot be executed within their optimal window, which means they cannot run with the finest time granularity in real-time.

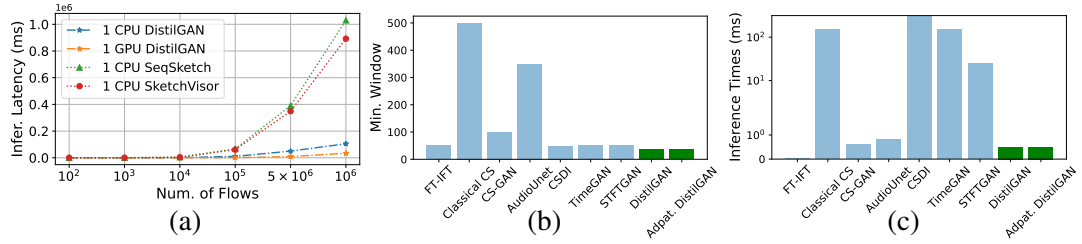


Figure 5.15: (a) Recovery Time with A100 GPU and CPUs for sketching based methods with different flow number; (b) Minimum (optimum) recovery time window size to achieve the best compression ratio with deep learning based models; (c) Inference latency under optimal window size.

Method	4×			16×		
	NMAE↓	NWD↓	NSWD↓	NMAE↓	NWD↓	NSWD↓
DISTILGAN	0.006	0.004	0.007	0.011	0.008	0.012
No Adaptive Sampling	0.007	0.007	0.009	0.016	0.009	0.018
No Learned Gaussian	0.008	0.012	0.030	0.016	0.018	0.048
No Temporal Generator	0.017	0.015	0.012	0.024	0.024	0.018
No Spectral Generator	0.011	0.01	0.035	0.017	0.025	0.088
No GAN Loss	0.016	0.024	0.066	0.025	0.047	0.089
No Auto-regression	0.01	0.009	0.014	0.015	0.012	0.023

Table 5.5: DISTILGAN ablation study results using the 5G RAN dataset.

5.6.4 Ablation Study

We conduct ablation study to quantify the contribution of different part of DISTILGAN to the high-fidelity reconstruction, and give more insights to the model design. In ablation study we focus on 5G RAN dataset for following reasons: (1) 5G dataset has the smallest granularity among all scenarios. which means this case is more challenging than IoT smart metering; (2) Evaluation with the NSWd and NWD metrics on ISP network flow size measurement is not straightforward as we do not care about spectrum in such scenario.

Comparison with baselines earlier in §5.5, has already highlighted the limitations of alternative designs. Here we examine the benefit from some of the key design choices underlying DISTILGAN through an ablation test in Table. B.3. Overall, all the key components play critical roles in the final fidelity. The adaptivity brings more gain when compression ratio is larger as there is more space to carry out different sampling rates. Specially, Adversarial Training (GAN loss), Time ResUnet (temporal thread), and the auto-regression structure contribute most to NMAE, while Learned Gaussian Noise and GAN loss jointly enhance the fidelity on temporal stochastic variation significantly. GAN loss plays critical role among all domains, the cannot show any gain compared with the other generative models without GAN loss. Although the spectral thread is designed to generate the high frequency and periodical features, as well as the spectrum, it does not perform well without GAN loss and Learned Gaussian Noise. Auto-regression structure for batch generation helps all the fidelity metrics evenly by providing information from the past.

5.7 Downstream Use Cases of DISTILGAN

5.7.1 ISP Network Microburst Detection

We evaluate the ability to detect microbursts at collectors when only relying on samples of traffic sent from network nodes. This is a common scenario when a network node has very limited resources and cannot afford any type of processing. Here, we define a microburst as a flow (or aggregate of flows) that exhibits the bitrate higher than a given threshold in a very short time window (microsecond-scale) [72].

We take a CAIDA ISP network trace [25] and recorded all the microburst flows using a time window of $1ms$ to obtain our groundtruth. We then sampled the trace at $10ms$ timescale to obtain a dataset of sampled traffic. We then recover the original traffic trace from the samples and use those to perform the microburst detection task. The F1 score is showed in Figure 5.16, DISTILGAN has significantly higher F1 score, especially when adaptive mechanism is activated (i.e., the sampling threshold is dynamically selected by our system). We also measure the average distance to the nearest real microburst to show the accuracy on temporal domain in Figure 5.17, and the results show that DISTILGAN has $1 \sim 2$ order closer distance to the real burst. Hence even DISTILGAN report a wrong detection, it will be very close to the real microburst.

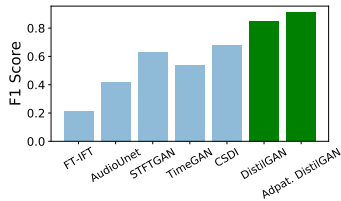


Figure 5.16: F1 Score of 99% percentile microburst detection.

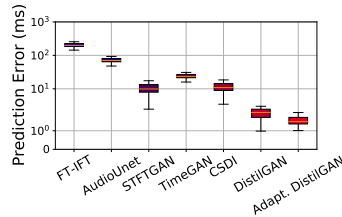


Figure 5.17: Average detection error to nearest microburst.

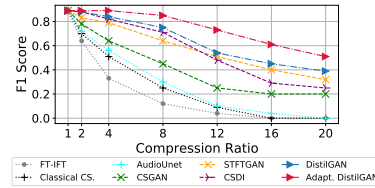


Figure 5.18: F1 score vs. compression ratio for O-RAN anomaly detection.

5.7.2 O-RAN Anomaly Detection

Here we started from data taken from a carrier grade Open RAN (O-RAN) testbed. We then used TranAD [170] software to detect anomalies present in the data and build our ground-truth⁵. Noteworthy that TranAD reports all the anomalies with a timestamp associated to them.

We then sampled the data using a fixed sampling rate similarly to the use case presented in §5.7.1. The obtained results are illustrated in Figure 5.18. Adaptive DISTILGAN can achieve $8 \sim 10$ CR without significant loss on F1 score, whereas the other generative methods can only makes it to $4 \sim 6$ CR. Among the non-adaptive methods, DISTILGAN stills gives the best performance at any CR. The non-generative methods cannot reserve any anomalous pattern when sampling rate is very low as they cannot easily learn those patterns.

⁵Anomalies here can happen because of radio interferences.

5.7.3 IoT Smart Metering Analysis

We started from a dataset including reports from four real smart meters [34]. This includes the temporal evolution of different variables such as power and energy consumption (Table 5.4). We used this as our groundtruth. We then sampled the data using different CRs and tried to reconstruct the original dataset with DISTILGAN. We compared this against AMDC [34], a lossy streaming-based compression/decompression algorithm specifically created for IoT smart metering reporting. In particular, we used AMDC to first compress the time series of data and then re-generate it.

The evaluation results are shown in Table 5.6. Here the two methods are compared in terms of NRMSE (Normalized Root Mean Square Error) of the reconstructed series with respect to the groundtruth. If the value is bigger than 0.2 then the reconstructed series cannot be considered acceptable [34]. As we can see, DISTILGAN can even reach a CR of 30 and still being able to retrieve the original series of data with a very small NRMSE. In contrast, AMDC fails beyond $CR = 4$. Finally, in Figure 5.19, we report the contribution to the Q-value for each considered attribute when using DISTILGAN with adaptive sampling. The weight of Apparent Power, Power, and Current increases with CR. This reflects the adaptive sampling mechanism can adapt to different variate as well.

Method	DISTILGAN	Adapt. DISTILGAN	AMDC
CR=30	Fail	0.059	N/A
CR=20	0.053	0.032	N/A
CR=16	0.051	0.029	N/A
CR=8	0.040	0.025	N/A
CR=4	0.032	0.019	N/A
CR=2	0.010	0.006	0.006

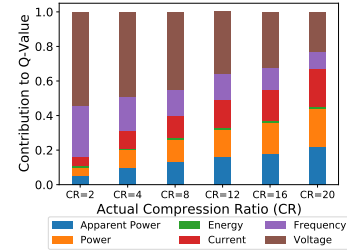


Table 5.6: Comparison with Streaming Figure 5.19: Contribution of each KPI Compression Algorithm in IoT system in the final Q-Value based on NMAE AMDC on Overall NRMSE (*Success if* when carry out adaptive sampling in $NRMSE < 0.2$ for each window, *otherwise* IoT smart metering. *fail*).

5.8 Summary

In this chapter, we investigate various network telemetry methods and propose our resolution — DISTILGAN, to realize a high fidelity, efficient, versatile, and real-time network telemetry system. DISTILGAN leverage a spectral-temporal-based generator with a reconstruction quality estimation-based method to achieve adaptive sampling/recovery network telemetry data. We also tailor the model to have highly parallel structure and optimize the inference process with GPU, achieving millisecond-level inference latency that can meet the latency requirement of real-time control loop. We evaluated DISTILGAN in three distinct scenarios, as well as different downstream use cases, and the results show a significant advance of DISTILGAN in the four requirements stated above.

Chapter 6

SpotLight

6.1 Introduction

In this chapter we investigate the explainable anomaly detection in networking. So far, there are existing works for general multivariate time series anomaly detection. However, most of them cannot give an explanation of anomaly detection, *i.e.*, highlight the root cause among all anomalous KPIs. We also find that the existing works are mainly designed for few open-source datasets and has never be evaluate on more complex system such as ORAN. Unfortunately, none of the existing method can give a satisfactory accuracy in ORAN system. The limited accuracy and lack of explanation motivate us to design and implement our solution for the anomaly detection in ORAN system.

The Open RAN architecture, with disaggregated and virtualized RAN functions communicating over standardized interfaces, promises a diversified and multi-vendor RAN ecosystem. However, these same features contribute to significantly increased operational complexity, making it highly challenging to troubleshoot RAN related performance issues and failures. Tackling this challenge requires a dependable, explainable anomaly detection method that Open RAN is currently lacking. To address this problem, we introduce SPOTLIGHT, a tailored distributed deep learning method running across the edge and cloud. SPOTLIGHT takes in a diverse, fine grained stream of radio network and platform metrics from the Open RAN system and the platform, to continually detect and localize anomalies. It introduces a novel multi-step generative model that allows us to detect potential anomalies at the edge using a light-weight algorithm, followed by anomaly confirmation and an explainability phase at the cloud, that helps identify the minimal set of KPIs that caused the anomaly. We extensively evaluate SPOTLIGHT using the metrics collected from an enterprise-scale Open RAN deployment in an indoor office building. Our results show that compared to a range of baseline methods, SPOTLIGHT yields significant gains in accuracy (13% higher F1 score), explainability ($4\times$ reduction in the number of reported KPIs) and efficiency ($4 - 7\times$ bandwidth reduction).

In summary, we make the following key contributions in this chapter:

- For the first time, we draw attention to the problem of anomaly detection and localization in the Open RAN context, highlight its uniqueness and the new challenges that need to be addressed.

- We introduce the SPOTLIGHT system architecture and method design to resolve the above problem. SPOTLIGHT employs a novel custom-tailored pipeline powered by a pair of deep generative modeling based anomaly detection methods, working in tandem and optimized for a distributed deployment, followed by causal discovery for enhanced explainability (§6.3).
- We develop a detailed and holistic Open RAN data collection process spanning both the radio network and platform dimensions, and we create, to our knowledge, the largest and most realistic multi-UE Open RAN dataset to date (§6.4). We commit to making the dataset publicly available within 6 months of publication of this work.
- We evaluate SPOTLIGHT on a realistic 5G RAN deployment and demonstrate its accuracy and explainability benefits compared to state-of-the-art solutions over synthetic anomalies, as well as its ability to detect and localize real world anomalies during normal RAN operation (§6.5-§6.7).

6.2 System Overview

6.2.1 Open RAN Architecture

Traditional RAN deployments are typically developed as embedded systems, where the hardware and software components are built by a single vendor and are tightly integrated.

Open RAN is an industry transformation, similar to SDN, that seeks to decouple RAN software (SW) from hardware (HW) and open up the interfaces between different SW components. It allows independent evolution of hardware and software, and faster rollout of new services. It also enables operators to mix and match their components through new and open interfaces, thus helping the ecosystem diversification.

As shown in Figure 6.1, an open and virtualized RAN base-station consists of several components. One is a radio unit (RU), deployed at a cell tower. Connected to it is a virtualized distributed unit (vDU), performing latency-critical operations, such as signal processing and radio resource scheduling. It runs on commodity servers, optimized for low latency (e.g. using Linux with real-time kernel patches). Due to stringent latency requirements, it is deployed at a far-edge site, at most within a few kms from the cell towers. One vDU can serve several RUs, depending on the HW capacity at the edge. Several vDUs connect to a virtualized centralized unit (vCU), which often runs at a near edge, further away from the towers, since it has more relaxed latency requirements. A typical large telco may have 10,000s of far-edge and 100s of near edge sites.

The Open RAN architecture standardizes the control and data plane through open interfaces. Entities like a service management and orchestration framework (SMO) and a radio intelligent controller (RIC) allow operators to control various aspects of RAN deployments, such as switching off RUs for power saving and optimizing handovers between cells.

In contrast to conventional RANs, Open RAN deployments are composed of several hardware and software components that are developed by different vendors and

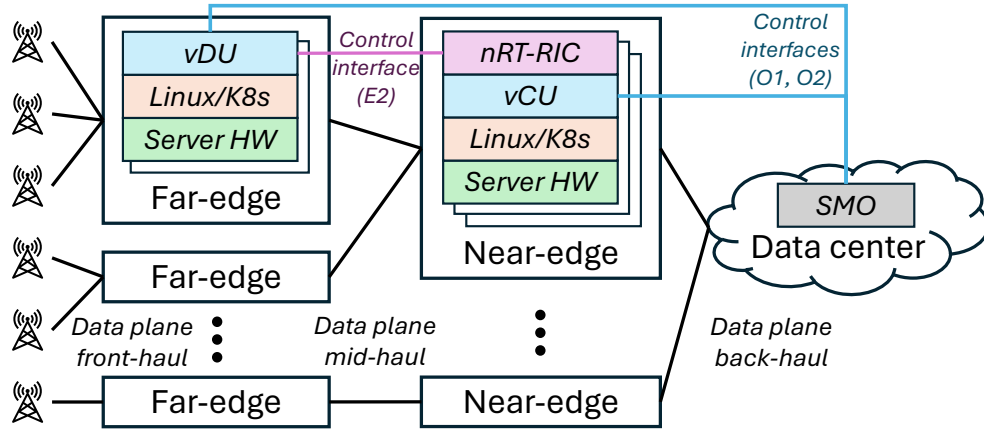


Figure 6.1: Open RAN defines various open interfaces and consists of hardware and software components provided by diverse vendors (marked in different colors). The RAN is deployed over 10,000s of far-edge and 100s of near-edge sites.

are bundled into a single solution by third-parties (e.g., system integrators). For example, a vDU and vCU from one vendor can run on server hardware from another vendor and the platform software from a third vendor. Different platform providers may use different versions of Linux and Kubernetes as a part of the platform. A key goal of Open RAN is to standardize different interfaces and architectures to enable such diverse deployments.

6.2.2 Key Challenges in ORAN Anomaly Detection

Massive Data Volume and Number of KPIs

The foremost challenge we face involves managing the substantial throughput of measurement data and the extensive array of Key Performance Indicators (KPIs). Identifying anomalies within hundreds of KPIs, each with a granularity of approximately 100 milliseconds, presents a significant hurdle. Effective anomaly detection must not only process individual KPIs but also discern the interconnected correlations among all KPIs. It is crucial to highlight the intricate nature of these system KPIs; they comprise a complex mix of various types, including categorical and binary, among others. This diversity and complexity render most traditional methods inadequate, often failing to yield reasonable results. Our approach, therefore, must be sophisticated enough to navigate and interpret this multifaceted data landscape effectively.

Realtime and Limited Resources

Real-time anomaly detection in Open RAN systems necessitates that all processes are completed swiftly while the system is operational, underscoring the need for rapid inference latency. However, reducing this latency isn't as straightforward as simply increasing the number of GPUs. This is particularly relevant in the context of far-edge implementations of the RAN system, where the use of GPUs is often not feasible due to budget constraints. Conducting anomaly detection efficiently using only CPUs at the far-edge poses a significant technical challenge.

Meanwhile, on the cloud side, while computational resources are typically more abundant, another issue emerges. Frequent data exchanges between the far-edge and the cloud can lead to substantial bandwidth consumption. This raises a critical question: how do we balance the communication overhead with the accuracy of the anomaly detection? Achieving this balance requires a thoughtful and well-designed approach, considering both the limitations of far-edge computing resources and the bandwidth constraints of cloud communication. Addressing this challenge is key to developing an effective and efficient anomaly detection system in Open RAN environments.

User Experience: Low False Detection

In the realm of anomaly detection research, there's a notable trend towards metrics-driven evaluation, with many studies primarily focusing on the overall F1 score. This score combines precision and recall, essentially evaluating the proportion of accurately identified anomalies. While this approach is valid for certain research problems, it may not align with the practical concerns of network operators.

From a network operator's perspective, the incidence of false positives – or incorrect anomaly detections – poses a significant challenge. Frequent false alarms can overwhelm operators, making it difficult to efficiently troubleshoot genuine issues. Therefore, a crucial aspect of our research is to enhance the precision of anomaly detection. Improving precision will reduce the rate of false positives, ensuring that anomaly reports are reliable and actionable. Addressing this issue is vital for developing an anomaly detection system that is not only effective in theory but also practical and user-friendly in real-world network operations.

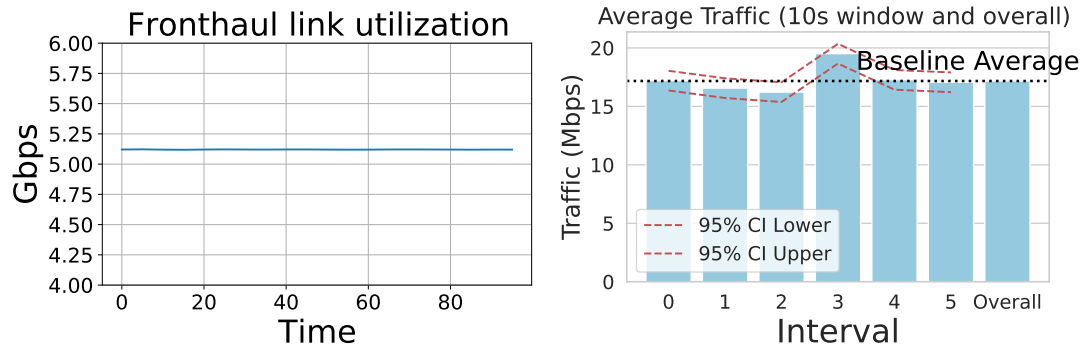
6.2.3 Explanation in ORAN Anomaly Detection

We would pursue certain explainable functions of the anomaly detection results, so that the user can spend less time on debugging. In this chapter, the term 'explanation' specifically pertains to elucidating the outcomes of anomaly detection, rather than detailing the operational mechanics of deep generative models. To clarify, this involves focusing on a concise list of potential root causes whenever an anomaly is identified. Such an approach significantly streamlines the user's task by reducing the need to examine a multitude of irrelevant KPIs. By concentrating on this targeted list, users can efficiently isolate and address the fundamental issues leading to the detected anomalies. This method of explanation not only enhances the practical utility of our anomaly detection system but also makes it more user-friendly by simplifying the diagnostic process.

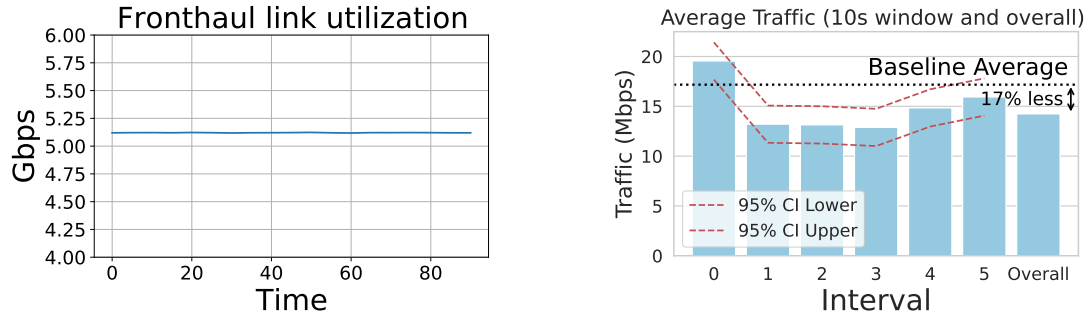
6.3 Overview of SPOTLIGHT

6.3.1 System Architecture

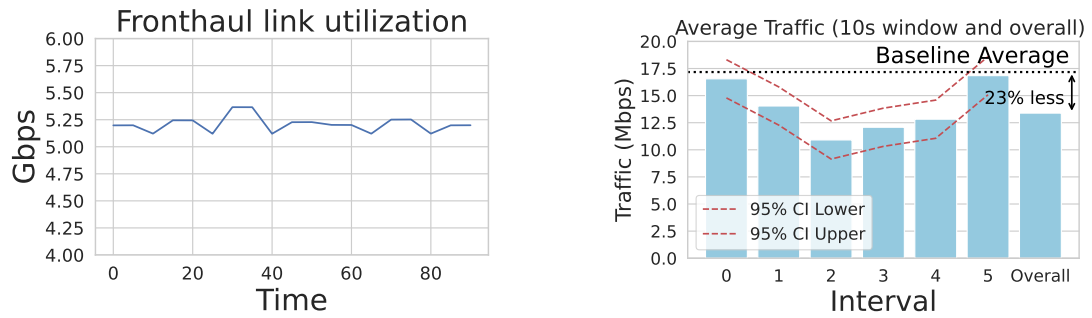
SPOTLIGHT is a system for detecting and explaining anomalies in RAN and platform components of Open RAN. Its primary focus is on vDU components which are most difficult to operate due to real-time operational requirements, but it is also applicable



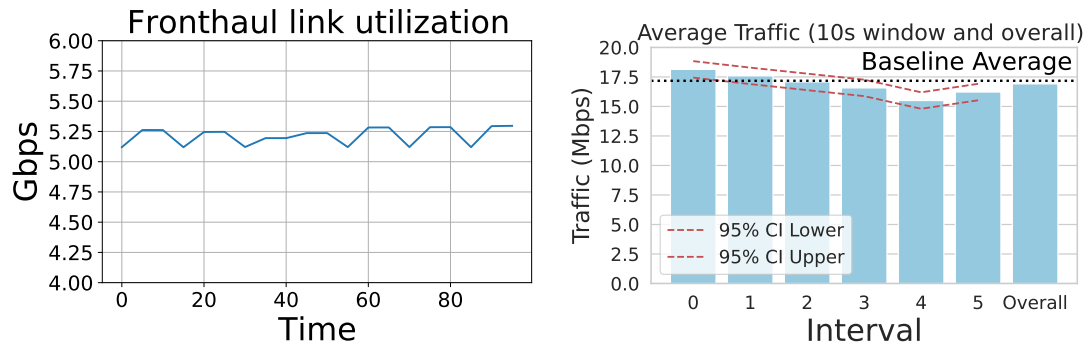
(a) Baseline KPIs for TCP DL Traffic



(b) KPIs for TCP DL Traffic with external radio interference



(c) KPIs for TCP DL Traffic with FH network contention



(d) KPIs for TCP DL Traffic with inadequate FH network contention

Figure 6.2: FH traffic and UE throughput for different scenarios

on other RAN components. It is built on observations that (i) anomaly detection needs to be automated as much as possible due to the expected scale of the RAN deployments in practice, and (ii) that in order to build this automation we need to collect detailed metrics from both RAN and platform. SPOTLIGHT is distributed across cloud and edge. Its high-level architecture is shown in Figure 6.3. It consists of three parts:

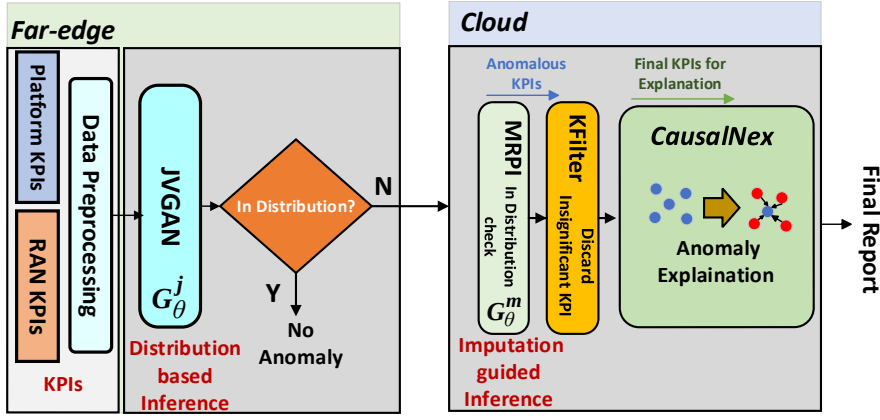


Figure 6.3: Schematic of SPOTLIGHT in the Open RAN context.

- **Data collection.** Unlike conventional RAN systems, SPOTLIGHT introduces detailed instrumentation on both the RAN and platform. We introduce probes that collect fine-grained metrics or KPIs (over 600 in total) that provide required data to enable reliable anomaly detection and root cause identification (see §6.4).
- **Data processing at the edge.** A far-edge node has limited compute capacity and is not suited for sophisticated ML methods. A centralized location (e.g., cloud), is better equipped for this task. However, sending massive amounts of data from 10,000s far-edge locations to a centralized place can be prohibitively expensive. We use the edge to implement a light-weight data filtering algorithm that draws a balance between the available compute and the uplink bandwidth.
- **Anomaly detection and identification in the cloud.** We deploy the heavy part of our processing in the cloud due to ample availability of compute and storage resources.

We collect data samples each 100 ms. We aggregate 50 consecutive samples into a window, representing multi-variate time series, and feed it to SPOTLIGHT’s anomaly detection method. At the output, in the event of an anomaly, we receive a filtered set of anomalous KPIs related to specific RAN and platform components that help easily identify the root cause.

6.3.2 Detection Method Description

We start with the design requirements of SPOTLIGHT based on the following requirements.

- **Accuracy.** Our goal is to maximize the detection of all anomalies (i.e., have a *high recall*) as well as minimize spurious detections or false alarms (i.e., have a *high precision*). To detect any kind of anomaly, we further require our method to be semi-supervised in that it should be trained only using normal data [29], and a limited amount of it. The method should also *generalize well* to new and unseen KPI patterns, including normal cases un-encountered in the training data.
- **Explainability.** We need an anomaly detection method that reliably directs us to the minimal subset of KPIs that point to the source of a detected anomaly. For

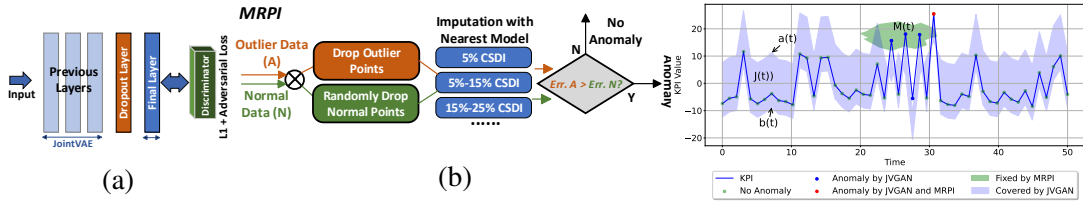


Figure 6.4: (a) JVGAN architecture; (b) MRPI workflow; (c) Illustration of learned distributions – $\mathbb{J}(t)$: blue; $-\mathbb{M}(t)$: green.

example, an anomaly in a physical (L1) layer of vRAN can be visible through unexpected uplink traffic variations in L1. But this will also cause variations in uplink traffic in MAC, RLC and PDCP layers, and we will potentially mark these metrics anomalous as well. The goal of the filtering step is to find the right root cause (L1 in this case) through a minimal set of relevant metrics, by learning and tracking dependencies between metrics.

- **Efficiency.** We seek a method that strikes a balance between leveraging available local processing resources at the far-edges, to minimize the bandwidth and cost requirements for further processing at the cloud, while ensuring detection and localization of any anomalies within the desired timescales.

To address these requirements, we structure the detection method into 4 components, shown in Figure 6.3. The first component is a light-weight pre-processing algorithm that runs at the edge, called JVGAN, which is based on a variational autoencoder. The key contribution we make here is to improve the accuracy compared to off-the-shelf methods, given that we have to deal with bursty and multi-modal data. JVGAN is also light-weight (takes less than 0.1% of a CPU core) and its output is significantly lighter than its input ($4 \times - 7 \times$), addressing the efficiency requirement (see Section 5.6.1).

JVGAN may have a high level of false positives (low precision). So we further process the data in the cloud to improve the accuracy. For that, we use an imputation based anomaly detector called MRPI. The idea behind it is that if we treat the outlier points from the JVGAN stage as missing points and cannot generate them with a time series imputation model (trained on normal data) then those outliers represent true anomalies otherwise false alarms. This allows to better capture noisy and bursty data. This step further improves precision by 5%-10% (see Section 6.6.1). Finally, we filter down the list of anomalous metrics to the minimal set. This is done in steps marked as KFILTER and CAUSALNEX in Figure 6.3, and addresses the explainability requirement.

The anomaly detection and localization problem we target takes as input the multivariate time series of KPIs $\mathbf{x} = (x^i(t))_i$, for each KPI i . This includes measured as well as derived KPIs, and span both radio network and platform. Resolving our problem translates to outputting $\emptyset$ if no anomaly is found, and a minimal subset $\mathbb{K} \subseteq \mathbf{K}$ otherwise, where $\mathbf{K}$ is set of *all* KPIs. In the anomaly case, the subset of KPIs $\mathbb{K}$ in the output reflects the likely cause and location of the anomaly, given that each KPI implicitly represents a location in the system. We continually perform the detection and localization for every incoming time window W of KPI streams. We next formalize the three key components of the algorithm.

Why use a generative model? The task of identifying anomalies in our study is framed as a binary classification problem, yielding deterministic outcomes (Yes or No). Labeling this process as ‘predictive’ underscores the definitive nature inherent in anomaly detection. Our choice to utilize a generative model, particularly one that employs probabilistic generation techniques, is primarily driven by the goal of distribution learning. The essence of our approach lies in learning a deterministic distribution through probabilistic means. In this context, if the data falls within the bounds of the distribution we generate, it is classified as non-anomalous. Thus, while the generative model serves as a tool for learning the distribution of potential values, the actual determination of anomalies is governed by a deterministic rule. This approach allows us to systematically categorize data as anomalous or not, based on a learned distribution that reflects the expected range of normal data variations. Besides, if learn in conventional way, with boundary and threshold, that would make the learning task more challenging since more independent parameters need to be learned.

Distribution based inference with JVGAN

As discussed in §2.5 and shown in §6.6.1, the prior RAN anomaly detection methods based on time series prediction or reconstruction are ineffective when applied to our setting. So we take a fundamentally different approach and create a new, generative modeling based anomaly detection method.

JVGAN, the initial component of our method, is a generator that first learns the distribution of normal KPI time series and then uses the learned distribution as a reference to infer if a given test KPI time series is anomalous. This distribution learning approach is not only robust to highly diverse patterns across many KPIs, as in our setting, but also does not have the threshold setting issue as prior methods. Furthermore, it has the beneficial effect of data filtering at the edge by efficiently distinguishing normal from potential anomalous cases so that only the latter cases are further inspected.

Specifically, for JVGAN we train a generator G_θ^j on the observed KPI time series $\mathbf{x}(t)$, where θ refers to the learned weights of the generator model. The objective of the generator G_θ^j is to infer whether or not a given observed KPI time series $\mathbf{x}(t) \in \mathbf{X}(t)$, which can be stated as:

$$\min_{\theta} \mathbb{E}_{t>0} [G_\theta^j(\mathbf{x}(t)) |_{\mathbf{x}(t) \notin \mathbf{X}(t)}] + \mathbb{E}_{t>0} [1 - G_\theta^j(\mathbf{x}(t)) |_{\mathbf{x}(t) \in \mathbf{X}(t)}],$$

where $G_\theta^j(\mathbf{x}(t)) = 1$ if $\mathbf{x}(t) \in \mathbf{X}(t)$, and 0 otherwise. The trained generator $G_\theta^j(\mathbf{x}(t))$ is then sampled N times to get a set of samples $\mathbb{J}(t)$ that approximates the true but unknown distribution of the ‘normal’ KPI time series $\mathbf{X}(t)$.

During classification, given a ‘test’ KPI time series $\mathbf{y}(t)$, we check to see if it falls within the learned distribution $\mathbb{J}(t)$. We use the upper and lower bounds of $\mathbb{J}(t)$, represented respectively as $a(t)$ and $b(t)$ that are calculated as the envelope (max and min respectively) of samples drawn from the distribution $\mathbb{J}(t)$, and we declare metric $\mathbf{y}(t)$ as anomalous if

$$\forall y_j \in \mathbf{y}(t), \exists a(t), b(t) \in \mathbb{J}(t) \text{ s.t. } y_j < b_j \text{ or } y_j > a_j \quad (6.1)$$

This is illustrated in Figure 6.4(c).

JVGAN generator $G_{\theta}^j(\mathbf{x}(t))$ is based on the variational autoencoder (VAE) [88], which is a standard approach for distribution learning. However, as we show in §6.6.1, vanilla VAE is ineffective for this purpose due to the following three issues:

1. Classical VAE works well only when dealing with continuous data, but there exist many KPIs in our setting, that are discrete or categorical (e.g., HARQ outcome).
2. Our KPI time series is highly bursty, which makes it harder for vanilla VAE to learn the distribution in a way that can reliably separate normal and anomalous cases.
3. Fitting the learned distribution too closely to training data hurts generalization, as training data does not represent all possible normal cases, i.e., $\mathbb{X}(t) \subset \mathbf{X}(t)$.

We address issue (1) in $G_{\theta}^j(\mathbf{x}(t))$ by considering JointVAE [44] as our basic neural network structure, as it is more robust with categorical and binary data streams. For issue (2), we include adversarial training (à la GANs [59]) for high fidelity distribution learning. For issue (3), we use Monte Carlo Dropout [58] to have the learned distribution to be not limited by training data. The JVGAN generator architecture ($G_{\theta}^j(\mathbf{x}(t))$) with the above techniques is illustrated in Fig. 6.4(a).

Imputation guided Inference with MRPI

JVGAN can reliably detect normal cases when a given test KPI time series is fully within $\mathbb{J}(t)$. Considering that $\mathbb{J}(t)$ is only an approximation of the unknown true distribution of the normal KPI time series, $\mathbf{X}(t)$, and that it is limited by the training data $\mathbb{X}(t)$, we have many cases that only fall partially in $\mathbb{J}(t)$. Fig. 6.4(c) illustrates such a case where some of the data points in the time series for a KPI fall outside the distribution $\mathbb{J}(t)$. However, simply inferring all such cases as anomalies will lead to poor precision (i.e., many false alarms). So, we introduce a further vetting step through another generator called multiple rate probabilistic imputation (MRPI) to minimize spurious anomalies.

The key idea behind MRPI is as follows.

The data points in a test KPI time series $\mathbf{x}(t)$ that fall outside $\mathbb{J}(t)$ are treated as ‘missing points’, and then we assess if they can be generated by an imputation model trained using only normal data $\mathbb{X}(t)$. Among those missing points, the ones that cannot be reliably generated through imputation, i.e., fall outside the distribution $\mathbb{M}(t)$ learnt by the imputation model, can be safely inferred as anomalous.

Our MRPI design is based on CSDI [162], which is the best existing time series imputation model [9]. However, the original CSDI design is limited to just one setting of missing data rate. On the other hand, it is impractical to have a separate imputation model for each possible missing data rate. We empirically observe that imputation at 10% intervals provides good generalizability within each interval. Therefore, we train multiple CSDI models for different intervals (ranges of missing data rates) using normal training data $\mathbb{X}(t)$: $\leq 5\%$, $5 - 15\%$, $15 - 25\%$, .. (11 models in total). During inference, we pick the model that is closest, based on fraction of data points in $\mathbf{x}(t)$ not covered by $\mathbb{J}(t)$. For example, if 10% of $\mathbf{x}(t)$ is not in $\mathbb{J}(t)$, then we pick the trained

5 – 15% CSDI model. We declare the point as an anomaly, if it does not belong to the distribution of the selected CSDI model. We illustrate this in Fig. 6.4(c) where the CSDI distribution is shown in green. The overall MRPI architecture is shown in Fig. 6.4(b).

Enhancing Explainability with KFilter and Causal Discovery

By applying the combination of JVGAN and MRPI, we can continually detect any KPIs exhibiting anomalous behavior based on the most recent batch of KPI data streams. However, the anomalous nature of some of these KPIs may be transient while other anomalous KPIs might be the effect of an anomaly caused elsewhere. So, to better identify the actual root causes of persistent anomalies, we employ two methods – KFILTER and Causal Discovery – as elaborated below.

KFILTER. The purpose of this method is to filter out insignificant anomalous KPIs. In particular, the aim is to discard those KPIs that are detected as anomalous only for a brief period of time but otherwise show normal behavior. So, we monitor the percentage of time each KPI is detected to be anomalous across the whole measurement period and filter out the ones which appear anomalous below a certain threshold period. We empirically set that threshold to 25% in our experiments.

Causal Discovery. Even after applying the KFILTER, there may be several anomalous KPIs left for a domain expert to examine to identify the root cause behind a detected anomaly. We observe that KPIs have inherent correlations and causal relationships between them. This suggests that the actual KPIs to inspect in the event of an anomaly are the subset of anomalous KPIs that have a causal relation from them to other anomalous KPIs. We aim to leverage the directed graph of causal relations among KPIs to reduce the ones reported by SPOTLIGHT in the event of an anomaly. Such an approach has proven to be effective in other settings such as root cause detection of failures in micro-services [74]. To deduce the causal graph among anomalous KPIs after KFILTER, we make use of CausalNex [16], the state-of-the-art toolkit for causal reasoning with Bayesian Networks. Specifically, each anomalous KPI is represented as a node in this graph and we report the ones that have directed edges from them to other anomalous KPIs.

Training

We train the JVGAN and MRPI models independently, as they focus on distinct tasks. We view the KPIs as different channels in the neural network, and therefore their inherent correlation is captured through weight sharing.

For both JVGAN and MRPI, we train the models by minimizing the Jensen-Shannon (JS) divergence between the distributions of the data and the corresponding models, following the conditional GAN formulations [73, 115] with the aid of a discriminator as in the GAN framework. We use L1 loss as part of the loss function for training. The discriminator of JVGAN is implemented with a single layer LSTM structure and a few dense layers, whereas in MRPI we keep the original discriminator as in CSDI [162]. Both JVGAN and MRPI have a loss function: $\mathcal{L} = \mathcal{L}_{L1} + \lambda \mathcal{L}_{JS}$, with a default weight $\lambda = 0.1$ to capture the effect of adversarial loss.

KPI Category	Description	Type
Signal Quality	Includes UL/DL SINR, CSI reports, UL/DL MCS	Radio
Packet Size	UL/DL packet size at PDCP, Midhaul, RLC, MAC and FAPI layer	Radio
Losses	UL Bler rate and DL HARQ NACK rate	Radio
Buffer info	UL Buffer status report and DL buffer occupancy	Radio
Resource Allocation	UL/DL PRB usage and UL/DL TBS	Radio
Thread Scheduling	Runtime of thread, count of on and off CPU, CPU ID	Platform
FH traffic	FH UL/DL link usage in Gbps	Platform
PTP logs	PTP frequency, RMS, delay and max offset	Platform

Table 6.1: KPI description

6.4 Data collection

One of the premises of SPOTLIGHT is access to detailed metrics. Here we describe the data collection process we have built as a part of the system.

Radio Network KPIs: We leveraged a powerful and lightweight telemetry extraction framework [56] that can collect detailed KPIs at a fine granularity and we integrated it in the RAN functions. This allowed us to collect many vCU and vDU events, such as signal quality reports of UEs, MAC scheduling decisions, queue and packet sizes at the PDCP and RLC layers, HARQ ACKs/NACKs, etc. Using the raw data, we derived several KPIs at the granularity of 100ms, including histograms and other statistics (min, max, distribution skewness, etc.). Using this process, we collected a total of 206 radio network KPIs, all agnostic of the number of UEs.

Platform KPIs: We collected network counters (number of packets and throughput) exposed by an API of the ToR switch at a granularity of 5s. We developed an eBPF tool [45], that allowed us to collect detailed per-CPU runtime information from all the threads running in the system (RAN and others), by capturing all the scheduling events. The tool exposes both on- and off-cpu runtimes of threads, i.e., how long a thread ran before being pre-empted (on-cpu) and how long it waited until it was scheduled again (off-cpu). As these events can reach hundreds of thousands per second, we aggregated the thread runtime statistics at a granularity of 1s to guarantee the stability of the system, in a similar way as we did for the radio network KPIs. We also scraped OS logs to collect data about PTP synchronization (1s granularity). This resulted in the collection of 466 platform KPIs in total.

A brief summary of all the collected KPIs, statistics and the data collection process is listed in Table 6.1, while a detailed description of the dataset’s schema can be found in [11].

6.5 Evaluation Methodology

6.5.1 Evaluation setup

We have deployed a state-of-the-art, enterprise-scale 5G Open RAN deployment, covering a five floor office building, with two 5G base stations per floor. All the compo-

Servers (8×)	HPE Telco DL110 Gen 10; Xeon 6338N CPU
Accelerator	Intel ACC100 for LDPC coding
Ethernet NIC	Intel E810 4×25GbE NIC,
Ethernet switch	Arista 7050-CX3
Radio unit	Foxconn 4x4 RU; 100MHz at 3.5GHz
PTP grandmaster	Qulsar Qg2 multi-sync gateway
Operating system	Real-time CBL-Mariner Linux kernel 5.15
PHY software	Intel FlexRAN v22.03
L2+ software	CapGemini 5G Solution

Table 6.2: 5G RAN hardware and software configuration.

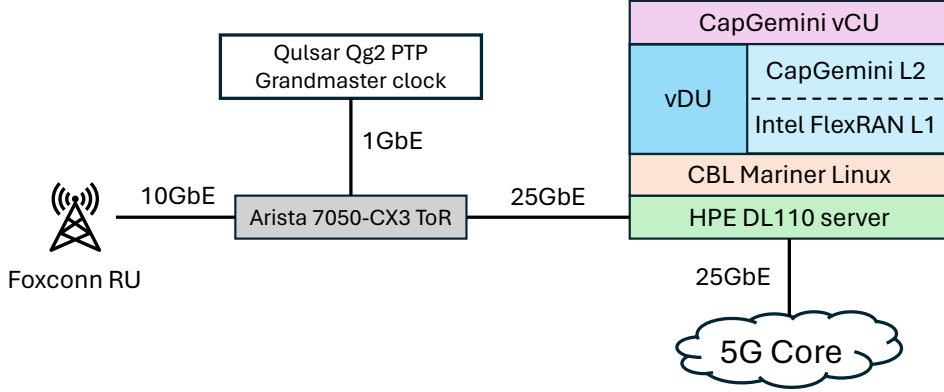


Figure 6.5: Deployment configuration for single cell.

nents are commercial-grade and O-RAN compliant. Table 6.2 summarizes the hardware and software configuration. The vRAN functions support several 5G features, including 100 MHz transmissions, 4×4 MIMO and the O-RAN 7.2x FH protocol. To our knowledge, this combination of features in a standards-compliant and end-to-end 5G Open RAN deployment is a unique characteristic of our deployment. We operated the deployment for over a year, and we identified a number of real-world cases of anomalies (discussed in Section 6.7.1) from this experience.

For the evaluation, we focus on a single cell of the deployment, using the configuration illustrated in Fig. 6.5. In terms of the end-user devices, we use up to 8 UEs during the network’s normal operation, including both commercial 5G smartphones (OnePlus N10, Samsung Galaxy A52s), as well as Raspberry Pi development kits equipped with Qualcomm 5G modems.

6.5.2 Dataset Creation

Using the deployment described in Section 6.5.1, we created a rich dataset consisting of fine-grained platform and radio network KPIs described in Section 6.4, considering both non-anomalous and anomalous cases. Overall, our training dataset consists of approximately 77 million measurement points. To our knowledge, this dataset is the largest and most realistic Open RAN dataset to date, with existing alternatives (e.g., [20]) mainly providing simulation/emulation data and not considering the real-time platform KPIs.

To generate a realistic dataset, we introduced a diverse set of eight traffic profiles, summarized in Table 6.3, and considered scenarios with one, five and eight UEs. In

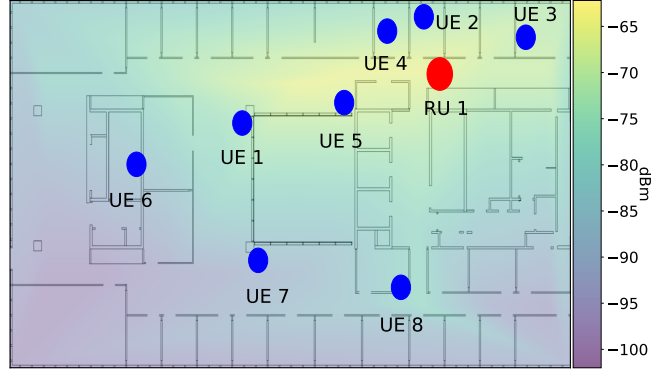


Figure 6.6: Floorplan of cell and UE deployment used for data collection and evaluation.

experiments involving a single UE, we generated traffic with all profiles for a duration of 10 minutes each. For five and eight UEs, we selected random UE subsets and created two scenario types:

- Constant traffic: Each UE selects a traffic profile and maintains it for 10 minutes.
- Mixed Traffic: Each UE randomly selects one of the 8 traffic profiles for a random duration (5 to 10 seconds). After this period, the UE pauses transmission for a random duration between 5 to 10 seconds before resuming with another randomly selected traffic profile. This lasts for 10 minutes.

Obtaining reproducible data in a controlled manner is challenging in a multi-UE setting with mobility. Therefore, in this work, we opt to use static UEs, meaning that some KPIs, such as SNR, are more static than in a truly mobile environment. We compensate for this in two ways:

- We consider several UEs with diverse positions and varying distances from the radio (showed in Fig. 6.6), ensuring that these KPIs differ across different UEs.
- All the KPIs of our dataset are aggregates across all UEs, rather than capturing individual UE metrics.

This combination, makes the aggregate signal quality related KPIs of the dataset dynamic. In addition to that, and due to (pseudo-)random traffic patterns that we introduce, all the other KPIs are also very dynamic.

6.5.3 Representative Anomalies

To evaluate the accuracy and explainability power of SPOTLIGHT, we carefully crafted a variety of anomalous scenarios. The anomalous scenarios under study were designed to cover representative issues across the vRAN stack, and are also discussed in §???. Specifically, we considered the following representative types of anomalies:

CPU contention – We emulate a class of anomalies in which thread contentions or misconfigurations are throttling threads that are responsible for RAN operations (see

Traffic Type	Discription	Duration (minutes)
iperf3 TCP DL	UE sends TCP DL traffic in DL direction	10
iperf3 TCP UL	UE sends TCP UL traffic in UL direction	10
iperf3 UDP DL	UE sends UDP traffic at 10 mbps in DL direction	10
iperf3 UDP UL	UE sends UDP UL traffic at 10 mbps in UL direction	10
file download	UE downloads the file from internet	10
file upload	UE uploads the file to server using scp	10
video stream	UE streams a video	10
web traffic	UE constantly access a random website	10

Table 6.3: Different profiles of generated traffic

Anomaly	Target	Effect	Duration	Frequecny	Ratio
PDCP worker thread contention	CU	Delays in processing of user packets, leading to packet drops	between 0.8 to 6 seconds	Every 10 seconds	10-65%
Radio interference	DU air interface	SNR reduction and packet retransmissions	between 1 to 2 seconds	Every 10 seconds	10-20%
FH network contention	FH and DU	SNR reduction and packet retransmissions	between 0.3 to 0.6 seconds	Every 15 seconds	10-12%
MAC scheduler thread contention	DU	Missed scheduling decisions and packet drops	between 0.8 to 1.2 seconds	Every 10 seconds	3-5%
Mixed anomalies	All	Combination of effects	Same as in single anomaly cases	Same as in single anomaly cases	varying

Table 6.4: Description of considered anomalies. The end result of all anomalies is UE throughput degradation.

Section 6.7.1 for a real-world example). We introduced a CPU contention using stressing [67] to a worker thread responsible for relevant processing (e.g. a PDCP worker thread for CU, etc).

Radio interference – This scenario reflects radio interference related issues that could be a result of inter-cell or external interference. For this anomaly, we configure a USRP software-defined radio to transmit an intermittent traffic over 40MHz of spectrum overlapping with our vRAN allocated spectrum.

Network contention – This scenario is meant to represent anomalies stemming from the sharing of network links between RAN and other functions, without proper isolation and QoS guarantees. For this anomaly, we introduced intermittent traffic on the same link that carries the FH traffic (IQ samples) between the DU and the RU.

Mixture of anomalies – In the mixed anomaly scenario, we created all six possible combinations of the four anomalies occurring together in pairs.

We summarize them in Table 6.4. Overall, we created a test dataset containing ~ 33 million datapoints. In Section 6.7.1, we further discuss several real anomalies that manifested in our deployment during its operation for over an year.

Category	Baseline Method
Prediction & Reconstruction	TranAD [170] VAE-LSTM [97]
Prediction based	GDN [41] LSTM-PRED [87, 169]
Reconstruction based	MADGAN [94] LSTM-AE [87, 169]
Statistical	Z-Score [5]

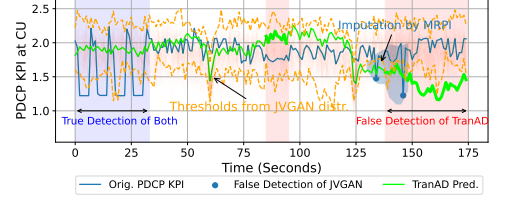
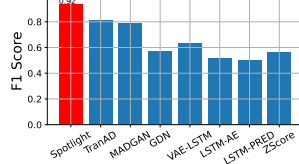


Table 6.5: Anomaly detection baseline methods.

Figure 6.7: Average F1 score across all scenarios.

Figure 6.8: Anomaly detection behavior of SPOTLIGHT vs TranAD.

6.5.4 Baselines

Accuracy baselines

As discussed in §2.5, most state-of-the-art time series anomaly detection methods in both RAN and ML domains are prediction based, reconstruction based or combine both. To assess accuracy benefit with SPOTLIGHT’s anomaly detection approach, we pick representative baselines from each of these categories, as outlined in Table 6.5. Like SPOTLIGHT, all these methods are trained on normal data.

Methods based on time series prediction (GDN [41] and LSTM-PRED [87, 169]) rely on the prediction error (i.e., difference between predicted and actual KPIs at each time step) for anomaly inference. On other hand, reconstruction based methods (MADGAN [94] and LSTM-AE [87, 169]) encode and decode the test time series input using trained models and infer anomalies based on the reconstruction error (i.e., discrepancy between decoded and actual test input). For LSTM-PRED and LSTM-AE methods, we use the standard deviation of prediction/reconstruction errors during training as the threshold for anomaly detection during inference.

As a simple-minded statistical baseline method, we also use a Z-Score based time series anomaly detection [5] in which z-score is continually computed over a sliding window and classify a new point in the time series as an anomaly if its z-score is above a threshold (one standard deviation).

Note that among these baselines, LSTM-PRED, LSTM-AE and Z-Score perform ‘univariate’ time series anomaly detection separately for each KPI, whereas SPOTLIGHT and rest of the baselines are multivariate across all KPIs.

Explanation baselines

SHAP with TranAD: SHAP [105] is a commonly used model agnostic method for explainability and provides a unified measure of feature importance – higher SHAP score for a KPI reflects its higher importance. We augment TranAD, the best performing anomaly detection method among our baselines, with SHAP (using its Omnixai [183] implementation).

GDN [41] has inherent explanation capability by modeling the set of KPIs (variables) as graph nodes and learning the edge weights between them through its attention mechanism. KPI(s) inferred to be anomalous and their highest weight neighbors forms the explanation output from GDN.

Besides the above two baselines, all univariate baseline methods we consider –

Method	PDCP			Radio			MAC			Network			Overall		
	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$
SPOTLIGHT	0.952	0.91	0.99	0.891	0.91	0.88	0.94	0.89	1	0.861	0.92	0.81	0.92	0.91	0.92
TranAD	0.88	0.78	1	0.82	0.75	0.90	0.89	0.80	1	0.70	0.62	0.80	0.81	0.73	0.92
MADGAN	0.80	0.70	0.91	0.78	0.73	0.82	0.82	0.69	1	0.75	0.65	0.88	0.79	0.69	0.91
GDN	0.72	0.56	1	0.73	0.58	1	0.51	0.34	1	0.32	0.19	1	0.57	0.41	1
VAE-LSTM	0.66	0.52	0.93	0.77	0.63	1	0.74	0.58	1	0.40	0.26	0.81	0.63	0.48	0.93
LSTM-AE	0.73	0.58	1	0.72	0.57	1	0.28	0.16	1	0.34	0.21	1	0.51	0.38	1
LSTM-PRED	0.71	0.55	1	0.72	0.56	1	0.26	0.15	1	0.31	0.18	1	0.5	0.36	1
ZScore	0.72	0.57	0.97	0.73	0.59	0.95	0.52	0.35	1	0.33	0.19	1	0.57	0.42	0.98

Table 6.6: F1 score, precision and recall for the 5 UE scenario and for all types of anomalies.

LSTM-PRED, LSTM-AE and Z-Score – provide explainability by default, as they treat each KPI independently and every KPI corresponds to a location in the Open RAN system.

6.6 Evaluation Results

6.6.1 Accuracy

We begin our evaluation by comparing the overall accuracy of SPOTLIGHT to the baselines from Section 6.5.4 for all the configurations identified in Section 6.5. As it can be seen from Fig. 6.7, the average F1 score of SPOTLIGHT for all the considered anomalous cases is 0.92, which is $\sim 14\%$ higher than the second best method (TranAD), demonstrating its high accuracy across all scenarios under study.

To understand the nature of SPOTLIGHT’s accuracy, we present a more detailed view of the results in Table 6.6, for all single anomalies in the case of 5 UEs with mixed traffic. The table presents F1 scores, precision and recall. We obtain similar results for all other scenarios, but we omit them due to lack of space. As we can observe, SPOTLIGHT has similar recall to the baselines, but fares significantly better in terms of the precision. This can be observed in the example illustrated in Fig. 6.8, in which we pinpoint the anomalies detected by SPOTLIGHT and the second best method, TranAD, for the PDCP worker thread contention anomaly during a 175s period. As we can see, in contrast to SPOTLIGHT, TranAD identified two false anomalies during 85-95s and 130-175s for the KPI under study. Instead, the JVGAN component of SPOTLIGHT correctly identified the samples during 85-95s as non anomalous, as they fell within the distribution it learned. On the other hand, JVGAN falsely identified the samples between 130s and 175s as anomalous. However, the imputation of MRPI corrected the false detection of JVGAN, flagging the KPI as non-anomalous, maintaining SPOTLIGHT’s precision at a high level.

The benefits of MRPI on improving the precision of SPOTLIGHT can also be seen in the ablation test of Table 6.7 for one of the 5 UEs scenarios. JVGAN captures most of the true anomalies, leading to a very high recall score for all anomalies under study. However, the precision of the model is fairly low, but is significantly enhanced by the introduction of MRPI at the expense of a marginally negative effect on recall. Finally, it should be noted that if we replace the JointVAE structure with a conventional VAE, then the precision becomes much lower, because the performance on categorical and binary variable is much worse. Introducing model uncertainty based tolerance contributes to the overall precision by reducing false detections. Without adversarial

Method	PDCP			Radio			MAC			Network			Overall		
	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	F1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$
SPOTLIGHT	0.967	0.94	1	0.92	0.95	0.89	0.95	0.91	1	0.91	0.99	0.85	0.92	0.91	0.92
JVGAN (MC Dropout)	0.95	0.90	1	0.87	0.80	0.95	0.90	0.82	1	0.84	0.84	0.85	0.87	0.83	0.95
JVGAN (W/o MC Dropout)	0.74	0.62	0.92	0.76	0.63	0.95	0.78	0.65	1	0.73	0.62	0.89	0.75	0.63	0.94
JVAE	0.67	0.50	1	0.70	0.54	1	0.72	0.56	1	0.67	0.52	0.95	0.68	0.53	0.97
VAE	0.65	0.48	1	0.66	0.50	1	0.57	0.40	1	0.43	0.28	0.90	0.51	0.36	0.97

Table 6.7: Ablation Test on 5 UE scenario

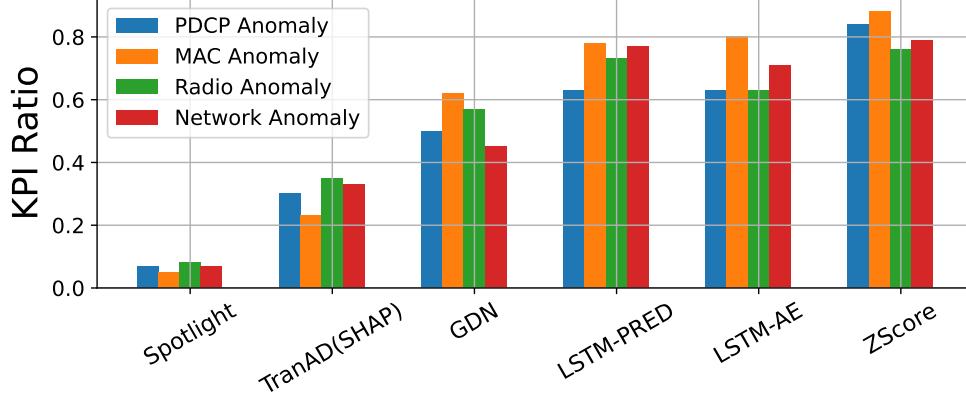


Figure 6.9: Ratio of KPIs flagged as potential causes for anomaly among all considered KPIs

training and MC dropout, the JointVAE does not perform better than other VAEs (*e.g.* VAE-LSTM), and therefore we can see that the adversarial training and MC dropout play critical roles in enhancing the precision.

6.6.2 Explainability

Here, we focus on evaluating SPOTLIGHT in localizing and explaining anomalies. We begin by comparing the explainability power of SPOTLIGHT compared to the baseline methods of Section 6.5.4. We first consider the ratio of anomalous KPIs that are flagged as potential culprits among all the considered KPIs for each anomaly by both SPOTLIGHT and the baselines. A low ratio indicates that a model is more focused and does well in filtering out irrelevant KPIs from the explainability step, effectively simplifying the root cause analysis process. Fig. 6.9 illustrates the ratios for all single anomalies that we considered. We observe that in all cases, SPOTLIGHT has a significantly lower ratio compared to the other methods.

Next, we zoom in the causal analysis results to better explain the benefits of SPOTLIGHT for explainability over the baselines. Here we use Anomaly Detection Ratio (**AD Ratio**) as a score for each anomalous KPI, reflecting their relative significance, after the causal discovery step of SPOTLIGHT.

Fig. 6.10 illustrates an aggregation of the KPIs that were flagged up by SPOTLIGHT and TranAD+SHAP for each considered anomaly, grouped by the category that the KPIs belong to. The height of each bar (y-axis) shows the anomaly detection score (or SHAP score in the case of TranAD+SHAP) of the most influential KPI of each category. The number of flagged KPIs of each category are shown at the top of each bar.

Taking as a concrete example the anomaly of PDCP contention in Fig. 6.10a, only 18 KPIs (out of more than 600) were flagged as potential culprits by SPOTLIGHT. Eight were localized to the platform layer and were related to the PDCP threads (the correct cause) and the rest to the radio KPIs of the RLC layer. In contrast, in the

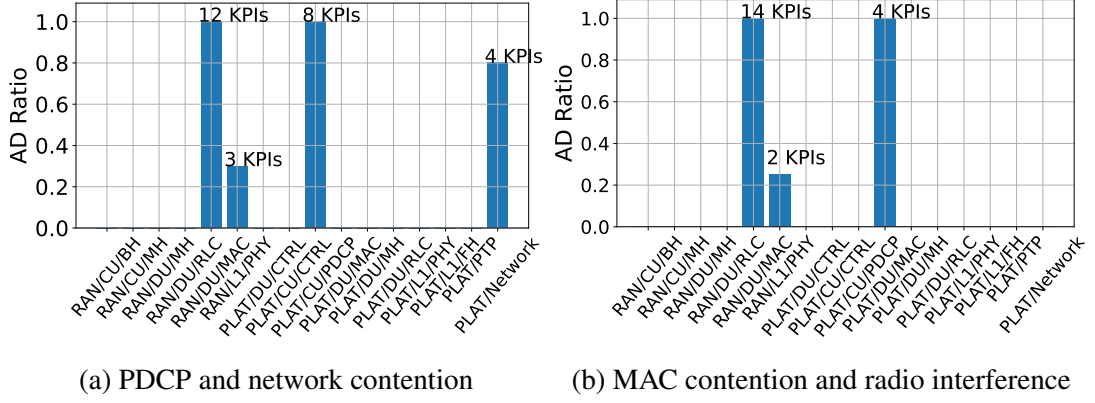


Figure 6.12: KPIs flagged by SPOTLIGHT for mixture of anomalies.

of MAC contention and radio interference. Similar observations about the causal detection capability of SPOTLIGHT can be drawn for all the remaining combinations of anomalies, which we omit due to lack of space.

6.6.4 Efficiency

We next show that our algorithm fits the architecture shown in Fig. 6.3 in terms of CPU and bandwidth requirements. We measure the time it takes to process metrics from one window of 5s on different computer architectures and we plot the ratio of the processing time over the measurement time (5s). If the ratio is 1 or below, the processing can be done in real-time, at least as fast as the data is collected. We perform measurements on an AMD EPYC 7453 28-Core CPU and NVIDIA RTX A5000 GPU(s). We show the results in Fig. 6.11. We see that JVGAN is well suited for running at the edge as it takes less than 0.1% of a CPU core to process the data at line rate. In contrast, MRPI requires almost 10 CPU cores or less than 6% of a GPU to run at line rate. That amount of compute power is not available at the edge, but can be easily accommodated in the cloud. Moreover, sharing 1 GPU (approx \$2,000) to manage 17 far-edge servers is a reasonable management overhead.

We next measure the network overhead. If we ship all the metrics to the cloud, the average required bandwidth is 1.5-2 Mbps per far-edge site, aggregating to 100 Gbps for a large telco network with 50,000 base stations, with very high ingestion and storage costs. When we run JVGAN at the far-edge and ship only its results to the cloud, the required bandwidth drops by 4 – 7 $\times$, a significant cost reduction.

Finally, JVGAN, MRPI, and KFILTER, also make the explanation more efficient by massively reducing the number of KPI to process, only 3% ~ 5% KPIs are used in explanation to show the anomalous KPIs with top significance. SPOTLIGHT only takes around 30 seconds to run CAUSALNEX for 10 minutes measurement. TranAD and SHAP combination fails to achieve this, requiring 4 ~ 5 $\times$ time of measurement period (*e.g.*, > 40 minutes to process 10 minutes measurement).

6.7 Real-world evaluation

6.7.1 Case Studies

In addition to the artificial anomalies that we introduced in Section 6.5.3, we used SPOTLIGHT in our real deployment to evaluate its detection and localization capa-

bilities with real, previously unknown anomalies. By collecting the exact same KPIs that we used for training, SPOTLIGHT allowed us to detect and troubleshoot three real anomalies in our network:

Intel ICE driver bug – This issue was linked to a misbehavior of the Intel ICE driver for the E810 NICs. Specifically, the driver version used in our deployment ignored the hints of interrupt thread affinities and placed interrupt threads to isolated CPU cores dedicated to the vRAN. This resulted in CPU contention with vDU worker threads, which degraded the performance of the vRAN. Once SPOTLIGHT successfully identify the thread contentions, it will led us to investigate the other processes that were running on the same cores, revealing the misbehaving threads. The issue was resolved, after upgrading the ICE driver to a version that introduced relevant patches [129].

PTP synchronization issue – This issue was related to a misconfiguration of our deployment, in which an NTP daemon (systemd-timesyncd) was running alongside the PTP daemon (ptp4l) that was synchronizing our vDU server with the Qulsar grandmaster clock. Every time that the NTP daemon would run, the server would get de-synchronized from the PTP grandmaster, resulting in detachments of the UEs and/or traffic drops. SPOTLIGHT identified the anomaly and successfully pinpointed the PTP timestamp KPIs as the culprit, which allowed us to quickly inspect the causes of the de-synchronization that disable the NTP daemon.

vDU RLC queue overflow bug – This issue was related to a bug of our vDU stack, that caused the RAN to drop all packets arriving at the RLC layer for all attached UEs, if saturated with traffic for a long time. SPOTLIGHT detected succesfully the anomaly and reported that the anomalous KPI was related to the RLC queues. After reporting this issue to our vRAN vendor, we were able to identify and rectify the bug.

6.7.2 Operational model

In this section, we demonstrate how SPOTLIGHT works in an operational environment. In such environment, it is important to have a very low fraction of false positives to reduce the unnecessary workload of operational teams. Our observation window is 5s (Section 6.3.1), which means that we have 12 observations per minute. We note that most of the RAN anomalies we had observed are not transient and last for many minutes. So in order to reduce the false positives, we aggregate the observations over one minute, and we report an anomaly only if we have 10 positive reports out of 12 windows. This means that the precision over 1h interval is $2 \cdot 10^{-4}$ and the recall is 0.91 – 0.98. We verify these results by running the system for 1h without an anomaly, and we note that we do not observe a single false positive during that time.

6.8 Discussion

6.8.1 Discriminator is not used to detect anomaly

In this chapter, the role of the discriminator is primarily focused on facilitating the adversarial training of the generators. To ensure a balanced training process, we intentionally designed the discriminator as a less powerful network. This approach helps in achieving consistent convergence during training. However, it's important to note

that this design choice limits the discriminator’s effectiveness in performing complex classification tasks.

Additionally, the discriminator is trained specifically to differentiate between the outputs generated by the generator and the actual ground truth data. This does not necessarily equip it to distinguish between normal and anomalous samples within real-world data. As a result of these limitations, the discriminator is not employed during the model inference stage in our setup. Its role is confined to the training phase, where it aids in refining the generator’s performance, rather than being directly involved in the final anomaly detection process.

6.8.2 How to understand the explanations

In this context, the term ‘explanation’ is used in contrast to scenarios where no explanation is provided, necessitating the review of all Key Performance Indicators (KPIs). Unlike a language model that translates detection results into simple, plain text, our explanations require domain-specific knowledge for interpretation. They are not straightforwardly converted into plain text but are instead presented in a format that necessitates a certain level of expertise in the field for full comprehension. The process of transforming these technical explanations into easily understandable plain text is beyond the scope of this chapter. Our focus here is on providing detailed, field-specific explanations that can aid experts in swiftly pinpointing and understanding the root causes of anomalies, rather than on simplifying these explanations for a general audience.

6.8.3 Choice of Models

We chose the JointVAE model as one of our foundational generators due to its robust performance with categorical data and its efficiency for CPU-based inference, making it a suitable choice for environments with limited computational resources. In terms of employing diffusion for data imputation, this decision was driven by its superior fidelity in time series imputation within our specific setup. While we have experimented with various other methods, only those that demonstrated notable effectiveness in different scenarios have been included as baselines in our study. This selective approach ensures that our comparisons are meaningful and grounded in real-world applicability, providing a comprehensive overview of each method’s strengths and limitations in various contexts.

6.9 Summary

In this chapter we investigate the efficient explainable anomaly detection in the ORAN system. We first carry out detailed measurement and data analysis on an ORAN test and introduce various contentions in different parts of the system. Our initial test shows that none of the existing methods can give a reasonable result on the anomaly detection, not to say explain the result. Therefore, we design and implement SPOT-LIGHT, an explainable anomaly detection system based on VAEGAN and the deep

diffusion model. Although high accuracy and sensitivity are achieved in anomaly detection, SPOTLIGHT also uses a lightweight generative model to filter out normal KPIs directly on the far edge. We also use a Bayesian network-based method — CausalNex to carry out causal discovery on anomalous KPIs. The evaluation result shows that our method has $4\times \sim 7\times$ better accuracy in explaining the root cause than the state-of-the-art.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we improve the performance of network data generation and monitoring with deep generative models. Through the different models used in this work, we prove that the deep generative model can enhance the fidelity and efficiency of most network monitoring tasks. We can also tailor the generative model to meet the real-time requirement in certain cases where there is no tolerance of monitoring delay. The exploration done in the thesis has a high versatility in the field of network monitoring, where we have tried our best to discuss as many as possible scenarios in network measurement, including ISP network, radio network, Kubernetes micro-services (ORAN), IoT system, etc. In addition to data generation and monitoring, we also successfully leverage a deep generative model to improve the explanation of network anomalies in the ORAN system. For each of the challenges stated in §1.2, we propose and implement the corresponding resolution, the high-level conclusions for each of these solutions are presented in the following subsections.

7.1.1 APPSHOT: Conditional Deep Generative Model for Synthesizing Service-Level Mobile Traffic Snapshots at City Scale

We have presented APPSHOT, a novel conditional deep generative model for synthesizing high-fidelity multiservice network traffic data that needs only publicly available context information of target regions. We have used real-world service-level mobile traffic data for multiple cities for our evaluation and show that APPSHOT not only outperforms a range of baseline approaches in terms of fidelity and also generalizes well to unseen regions. Our patch-based learning approach and corresponding operations have proved to be effective in generating traffic for cities with different sizes. Also, data augmentation with overlapping patches significantly enhances the performance with respect to handling traffic hotspots and diverse traffic ranges. The APPSHOT neural network architecture and the service level constraints it incorporates significantly enhance the accuracy of service compositions in synthetic traffic, while preserving a strong structural correlation between services. Furthermore, APPSHOT is shown to capture realistic statistical variations on both city-wide traffic demand and structural characteristics (e.g., number of hotspots). Finally, we have demonstrated the utility of

APPSHOT-generated data through a use case on radio network slicing.

7.1.2 GENDT: Mobile Network Drive Testing Made Efficient with Generative Modeling

We have presented GENDT, a new conditional deep generative model. GENDT is the first data generation method for radio KPI time series data, aimed at reducing the measurement effort with drive testing. It embeds a number of innovative aspects, including the use of stochastic layers on top of a GNN and LSTM based network to process dynamic input network context and to model stochasticity, and batch based training and generation for high fidelity long series generation. We evaluate GENDT with real drive test measurement data from two different countries, covering a wide range of scenarios. Our results show that GENDT generally outperforms a range of baselines, and by a big margin. We also show that GENDT can generate radio KPI time series over long and complex trajectories with high fidelity. Moreover, GENDT is being able to tell apart data uncertainty from model uncertainty. The knowledge of model uncertainty in turn enables selection of the most informative measurement data for model training, which can significantly reduce the measurement overhead — our results show the potential to optimize measurement efficiency by up to 90% that the efficacy of GENDT-generated data to support downstream drive test measurement use cases is comparable to that of real data.

7.1.3 DISTILGAN: High Fidelity, Efficient, Versatile and Real-Time Network Telemetry with Deep Generative Modeling

In DISTILGAN, we have tackled the significant network telemetry challenge of accurately and timely understanding network behavior while minimizing the data collected from various network elements. We further aimed for a method that requires minimal processing and memory resources at the sending nodes (i.e., just sampling at minimal rate) and thus has broad applicability including with commodity network switches and resource-constrained devices (e.g., IoT sensors). We presented DISTILGAN, a new data-driven solution based on deep generative modeling that meets all the essential requirements for a network telemetry method: high fidelity, efficiency, versatility, and real-time. DISTILGAN achieves high-fidelity network measurement data reconstruction through a custom-tailored deep generative model operating across temporal and spectral domains. It can also adapt to the minimal sampling rate required for specific fidelity objective on measurement nodes, by analyzing the reconstruction quality from received data stream through a novel Q-value measure. We evaluated our method using three real and diverse scenarios: (1) a large ISP network; (2) 5G radio access network (RAN); (3) IoT (smart metering) network. Compared to a wide range of baseline approaches representing the state of the art, we show that DISTILGAN yields significant gains with respect to all four of the above requirements as well as for representative downstream use cases. For example, DISTILGAN provides $25\times$ gain in efficiency for the same fidelity compared to state-of-the-art method for flow-level monitoring in ISP networks. Moreover, DISTILGAN achieves 1~2 orders of magnitude lower inference latency than prior art to support real-time monitoring and control applications at few

ms scale.

7.1.4 SPOTLIGHT: Accurate, Explainable and Efficient Anomaly Detection for Open RAN

In this work, we presented SPOTLIGHT, a tailored deep learning based method for Open RAN anomaly detection and localization, that is distributed between the edge and the cloud. SPOTLIGHT manages to provide highly accurate and explainable anomaly detection results, that are significantly better compared to state-of-the-art methods, while remaining computationally efficient. To train and evaluate SPOTLIGHT, we developed a large scale and realistic data collection process spanning both the RAN and the platform layer on an enterprise- scale 5G Open RAN deployment. As a future work, we are planning on evaluating SPOTLIGHT at a larger scale.

7.2 Future Work

This section summarizes the limitations and future work opportunities in relation to the contributions made in this thesis.

7.2.1 Network Data Generation

APPSHOT and GENDT cover the most common data generation task in networking, mainly in 1-D data (*e.g.*, univariate time series), 2-D data (*multivariate time series and traffic map in a period*), and 3-D data (*e.g.*, spatial-temporal distribution of traffic). Nevertheless, there are still scenarios that are not discussed in this thesis.

High Dimension Data generation. One example for high-dimensional data would be point-cloud data; a set of data points in a 3D coordinate system might change over time (then become 4D). Such data structure becomes more and more common in current network, such as Lidar system. There are two challenges we need to resolve for higher dimension data: (1) fundamental changes in data structure, 4D data is much more complex; (2) More challenging to be real-time, 4D data in general come with massive throughput but in many scenarios (*e.g.*, automatic driving), real-time processing is needed.

An Universal Framework. In this thesis, our solution is more about a tailored method for each scenario. It would be more efficient if the user could rely on a general framework for most of the $M - to - N$ generation task, where M is the input dimension and N is the output dimension. One alternative is using graph or point cloud to represent all data structures but automatically select the dimension to learn correlation between different inputs. This part will be left for future exploration.

7.2.2 Network Telemetry

The proposed method, DISTILGAN, can only work for data that can be represented as a general multivariate time series. In fact, measurement data could be topological or textual logs, which cannot be processed by DISTILGAN. For topological data, we can enhance the current version with extra design to accommodate categorical data, and

when the dimension goes beyond multivariate time series, we need to propose a new model for that. For textual logs, such as heads of IP packets, we might want to explore the application of NLP (natural language processing) methods to detect events hidden in 'normal' traffic.

7.2.3 Anomaly detection

In this thesis, we design and implement SPOTLIGHT for anomaly detection in the ORAN system. However, there are still pending tasks to explore in the future, which is not feasible for now due to the available experiment resources.

Multiple Cells. Because of the current implementation of the testbed ORAN system, we can only use one cell each time. In real world, we would expect at least tens or hundreds of cells for each cloud server. The implementation of multiple cells poses a challenge on the generalization of the proposed method among different cells, introduces more transmission overhead for measurement, and leads to a significant increase in workload on the cloud side. More challenges might be noticed once multiple cells are available.

Mobility. Our current setup is an indoor implementation of ORAN, obviously we cannot reflect the complex radio environment a user might experience in the real world. High mobility will cause significant fluctuation on radio KPIs such as SINR and makes the switch between different cells more frequent. The concept proposed in SPOTLIGHT needs more evaluation in the context of high mobility.

More System KPIs. Current SPOTLIGHT focuses mainly on RAN KPIs and ORAN system KPIs. In fact, the other measurements about the system can also reflect anomalies. For instance, the thread of ORAN might immigrate to different CPU cores because ORAN is deployed on top of Kubernetes. Such immigration might cause issues when two critical threads are moved to the same core. Our current setup of SPOTLIGHT does not consider the KPIs such as CPU utilization, etc., and therefore cannot capture the CPU thread immigration issue. In the future, we will try to include those extra KPIs in anomaly detection to detect more anomalous events of the ORAN system.

Appendix A

Work and Publications

A.1 Publication Related to APPSHOT

Chuanhao Sun, Kai Xu, Marco Fiore, Mahesh K. Marina, Yue Wang and Cezary Ziemlicki, "AppShot: A Conditional Deep Generative Model for Synthesizing Service-Level Mobile Traffic Snapshots at City Scale," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4136-4150, Dec. 2022, doi: 10.1109/TNSM.2022.3199458.

A.2 Publication Related to GENDT

Chuanhao Sun, Kai Xu, Mahesh K. Marina, and Howard Benn. 2022. GenDT: mobile network drive testing made efficient with generative modeling. In Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22). Association for Computing Machinery, New York, NY, USA, 43–58.

Appendix B

Implementation Details and Extra Results

B.1 Data Analysis and Model Details of GENDT

B.1.1 Visualization of Environment Context Attributes

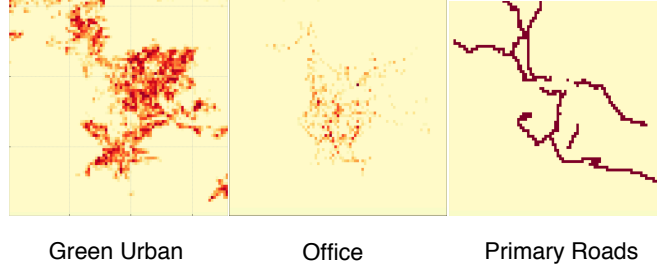


Figure B.1: Spatial distribution of 3 selected environment context attributes in DATASET B.

B.1.2 Details of Stochastic Layers

The intensity of noise is controlled by a function. When we add noise, we do not want to change the total value of hidden state of all hidden dimensions, so we have:

$$h'_t = (h_t + a_h n_{t,h}) \frac{\sum_{i=1:H} h_{t,i}}{\sum_{i=1:H} (h_{t,i} + a_h n_{t,h,i})}, h_t = \{h_{t,1}, \dots, h_{t,H}\}$$
$$c'_t = (c_t + a_c n_{t,c}) \frac{\sum_{i=1:H} c_{t,i}}{\sum_{i=1:H} (c_{t,i} + a_c n_{t,h,i})}, c_t = \{c_{t,1}, \dots, c_{t,H}\}$$

Where H is the dimension of hidden state h_t and c_t . Using different a_h and a_c , we can control the relative intensity of noise to the hidden states, and thus control the uncertainty level during training.

We use a different training method compared with [57], where the learning was done by variational inference with an inference network introduced to use the backward-recurrent state to approximate the nonlinear dependence of h'_t with future observation

Environment Context Attribute	
Land Use Type	PoIs
Continuous Urban	Tourism
High Dense Urban	Cafe
Medium Dense Urban	Parking
Low Dense Urban	Restaurant
Very-Low Dense Urban	Post/Police
Isolated Structures	Traffic Signal
Green Urban	Office
Industrial/Commercial	Public Transport
Air/Sea Ports	Shop
Leisure Facilities	Primary Roads
Barren Lands	Secondary Roads
Sea	Motorways
	Railway Stations
	Tram Stops

Table B.1: List of environment context attributes considered. See examples in Figure B.1

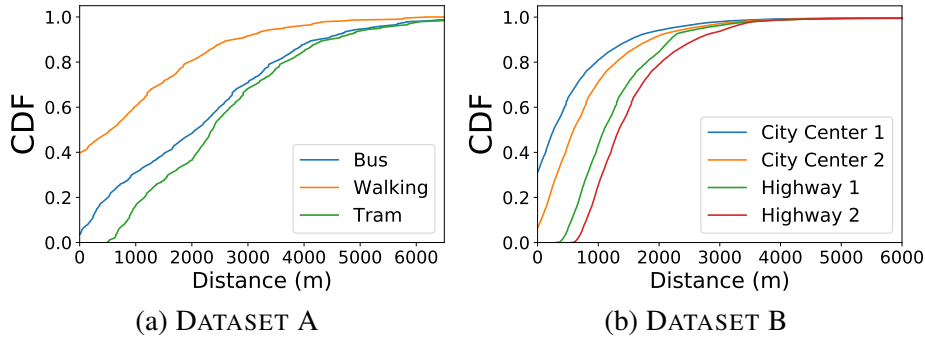


Figure B.2: CDF of distance to serving cell in different scenarios.

$x_{t:T}$ and states $h_{t:T}$. Instead, in our case effective training of SRNN is realized via adversarial training with a discriminator. A LSTM based discriminator provides extra training signal on top of the L2 norm loss function to make the model converge with nonlinear dependence of h'_t .

B.1.3 Hyper Parameters

We use single layer LSTM network for both GNN-Node and aggregation networks in the GENDT generator. Hidden layer dimensions for both GNN-Node and aggregation networks are set to 100.

We use 50 for the batch length by default and the default step length is set to 5. Note that, in our experiments, we found that any step length between 1 and 15 gives similar result.

Noise intensity $[a_h, a_c]$ are chosen in the range of $[1, 3]$ with the best fit of histogram – larger intensity means more significant variation in output series but needs to be fine-tuned per scenario. In general, $a_h = a_c = 2$ gives good results for most of the cases.

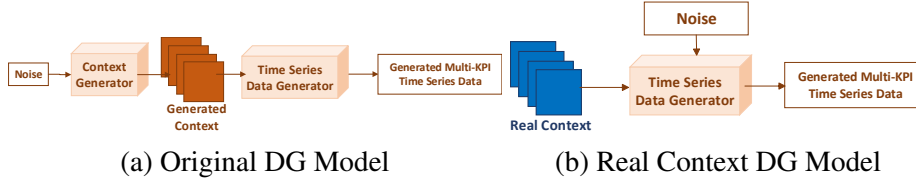


Figure B.3: Schematic of original DoppelGANger (DG) and its optimized variant.

B.2 Discussion on DoppelGANger

As DoppelGANger (DG) seeks to provide a generic data generation architecture across different types of time series data and use cases as well as allow hiding sensitive context (called metadata in DG), it adopts a two stage generation process. In the first stage, context is generated from noise through an unconditional GAN model. The generated context then is used to condition (control) the generation of target time-series network/system data in the second stage via a conditional GAN model.

From the perspective of our mobile network drive testing data generation problem and our proposed GENDT method, DG has four key limitations:

- The DG model architecture cannot handle dynamic network context input. GENDT overcomes this issue through a tailored GNN based generation model.
- There is very limited support for modeling stochasticity in DG via naive direct injection of noise as input to the model. GENDT, on the other hand, comprehensively and effectively deals with this issue through stochastic layers in the model as well as noise input through its residual generator.
- DG adopts a batch generation approach for long time series generation, while GENDT builds on this and optimizes it much further through its autoregressive residual generator and training with overlapping batches.
- DG lacks any mechanism to minimise training data required, whereas GENDT has the built-in residual generator to provide cues on the need for more training data.

It is worth noting that the motivation behind DG (and even SpectraGAN) is to overcome the barrier to accessing real data stemming from commercial sensitivity or privacy concerns, whereas the high cost of measurement data collection with drive testing motivates our design of GENDT.

B.3 Additional Evaluation and Use cases

B.3.1 Need to Support Long Series Generation

Note that for high fidelity drive test data generation, it is essential to support long series generation. To illustrate this point, we compare GENDT with two cases, where the data for the long (2200+s) target trajectory considered in this subsection is instead obtained by stitching data from multiple independently generated short (50s and 100s) trajectories. Results shown in Table B.2 clearly indicate that short trajectory generation

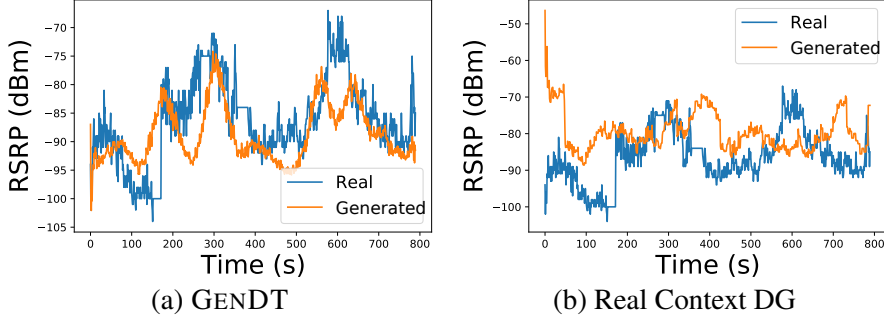


Figure B.4: Sample of generated RSRP time series with GENDT and real context DG in DATASET A for the Walk scenario.

Method	MAE↓	DTW↓	HWD↓
GENDT	11.69	7.18	10.4
50s Trajectory	14.50	10.1	18.79
100s Trajectory	13.11	9.05	16.86

Table B.2: GENDT performance compared with short trajectory generation for long trajectory case in DATASET B.

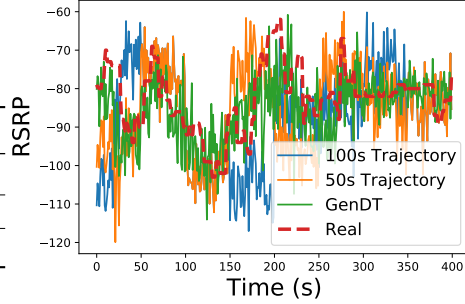


Figure B.5: GENDT-generated RSRP time series compared with short independently generated trajectories.

does worse than GENDT, especially in terms of the data distribution (HWD metric). Visualization of RSRP series generated with these alternatives (GENDT and 50s/100s short independent trajectories) in Figure B.5 clearly highlight the artifacts at the points successive short trajectories are stitched together, whereas GENDT-generated RSRP time series samples closely track the real measurement data. Note that in this figure, we zoom in on the last 400s of the long trajectory to allow the differences to be clearly seen. These results overall highlight the need to capture long-term temporal relations in the data to ensure high fidelity generation.

B.3.2 Ablation Study

Comparison with baselines earlier in §4.6.1 has already highlighted the limitations of alternative designs. Here we examine the benefit from some of the key design choices underlying GENDT through an ablation test. For this, we consider RSRP and RSRQ KPIs, common to both datasets, and report results with DATASET B.

From the results in Table B.3, we see that RESGEN plays a critical role in effectively introducing noise to help model stochasticity. Without RESGEN, GENDT degrades considerably in terms of the HWD metric. An interesting related observation is that environment context input through RESGEN in GENDT does not always help in improving the fidelity on other metrics (MAE, DTW), maybe because KPI dynamics can be high for the same input environment context. In contrast, the use of stochastic layers (SRNN) consistently improves all metrics, including HWD targeted by this

Method	RSRP			RSRQ		
	MAE↓	DTW↓	HWD↓	MAE↓	DTW↓	HWD↓
GENDT	8.10	5.89	7.67	1.7	1.34	1.65
No RESGEN	7.99	6.60	13.7	1.6	2.3	9.7
No SRNN	11.53	8.89	10.4	2.4	1.99	4.8
No GAN loss	14.66	12.45	15.3	3.8	3.6	6.9
No batch	12.9	9.60	10.5	2.6	2.3	3.7

Table B.3: GENDT ablation test on DATASET B considering RSRP and RSRQ.

mechanism.

Ablation results indicate that the adversarial training (i.e., use of discriminator) is key to GENDT performance. Dropping ‘GAN loss’ from the loss function results in the most performance degradation on all metrics compared to all other design choices. The adversarial network of GENDT is trained to learn to play a similar role as the Inference Network in [57], and thus it is critical for effective model training. As expected, the use of batch generation and training with overlapping batches has a beneficial effect on MAE and DTW fidelity metrics but also improves HWD. The batching related mechanisms are particularly effective when generating data for long trajectories, as previously highlighted in §4.6.2.

B.3.3 Further Use Cases

GENDT is intended to support the use cases that rely on traditional drive testing. We evaluated GENDT for two such cases in §4.8.2. Here we outline several more example use cases. While GENDT can be readily applied to these use cases listed below without reliance on drive test measurements, evaluating its effectiveness requires access to relevant KPI measurement data as well as ground-truth for use case specific metrics.

- *Video Streaming QoE Prediction.* Depending on the QoE metric, measurement of multiple radio KPIs are required to infer the video streaming QoE [114]. GENDT can support this use case along the lines of throughput and PER prediction use case we highlighted in §4.8.2.
- *Cell Load Estimation.* In [30, 141], the authors proposed using RSRQ and SINR to estimate the cell load under different scenarios. Since we do not have the ground truth cell load information, we are not able to verify the accuracy of these methods. But these prior works offer a way to infer cell load through drive test measurements, which can be efficiently supported with GENDT.
- *Link Bandwidth Prediction.* In [193], the authors identify five KPIs has significant correlation with link bandwidth (namely, RSRP, RSRQ, CQI, Handover, and BLER) and proposed a method to infer the link bandwidth with these five KPIs. As we have considered several of these KPIs, it would be straightforward to support this use case with GENDT and evaluate it when real link bandwidth measurement data is accessible.
- *Uplink Network Jitter Prediction.* KPIs such as RSSI, Cell ID, device location, RSRQ, RSRP and, importantly, the average transport block (TB) size, enable

prediction of uplink jitter [149]. This use case can be supported by GENDT via generation of data for these aforementioned KPIs.

What-If Analysis Studies. Over and beyond the type of radio KPI based use cases mentioned above, the context driven design of GENDT naturally lends itself to what-if analysis studies. An example of such a study is to examine the impact of deploying new cells in the operator’s network on radio KPIs, *prior* to deployment. Another example is to easily study the effect of recent/potential changes in the environment context of a target region (e.g., construction of new highways or big buildings) on radio KPIs without needing to conduct drive test measurements.

B.4 Configuration Details of DISTILGAN

B.4.1 Epoch Length

While evaluate on different epoch length, we actually reveal the robustness of different method on the input data distribution. As the example in Figure B.6 shows, there is a multi-modal distribution at 10ms scale, whereas such effect is not significant at 1 second. Also the tail at 10ms is much shorter than 1s in some time window, on this specific example, 10ms epoch has 1/30 of flow number of 1s epoch length, but the tail is 8×10^2 shorter than 1s case. All those observations point to one conclusion: to ensure the sparsity and orthonormality for Classical CS-based methods, we cannot simply scale down the number of counters by the expected flow number. Instead, we must introduce a certain level of redundancy.

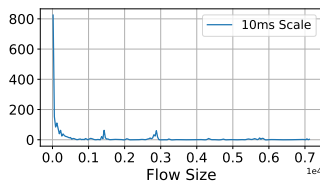


Figure B.6: PDF at 10ms scale.

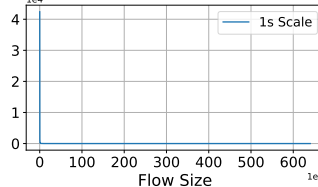


Figure B.7: PDF at 1s scale.

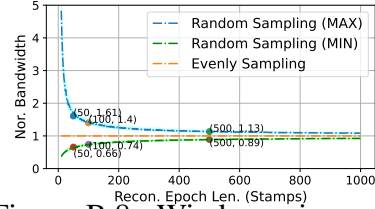


Figure B.8: Window-wise sampling rate with random sampling and periodical sampling

B.4.2 Memory Configuration

We evaluate DISTILGAN on top of SeqSketch [71] with CAIDA network traces [25], where the compressive sensing algorithm is replaced by DISTILGAN, and we take SeqSketch as a way to subsample the original input. Since SeqSketch’s mechanism has bias for the significant flows as they compete for hash table by flow size, we do not use extra threshold based sampling. As fore the memory size, considering the number of flows in each epoch, for one second epoch length, we set the a basic memory size at 128KB Fractional Sketch (FS), 96KB Key Value (KV), and 32KB Bloom Filter (BF), and adaptive on the FS size with a 48KB step. The maximum step number $n_{s,max}$ is 10, and the actual memory is configure to:

$$128KB(\text{Const. FS}) + n_s \times 48KB(\text{Adpt. FS}) + 96KB(\text{KV}) + 32KB(\text{BF}), n_s \in [1, 10]$$

During recovery process, instead of using current epoch measurement, DISTILGAN also takes in the results from previous epoch, so that DISTILGAN is able to **capture temporal pattern of each flow**. The different flow will be taken as different input channel of DISTILGAN generator. The memory size is scaled down with the number of expected flows following the configuration in §7.3 [71], which varies between different epoch length.

DISTILGAN will adjust n_s to adapt the $Q - value = 0.01$ or other tolerance as stated. When compute Q-value, we mask some records randomly by FS ratio to mimic the information loss due to limited memory, and keep using $N_s = 2$. n_s has initial value at 10 and then the model will learn to reduce it, and minimal value of n_s is 1 to make sure we can compute the Q-value. According to [71], the maximum memory size is more than sufficient to measure 100K level flows with classical CS. For longer epoch length, we increase the size of each part according to the average number of flows in each epoch accordingly.

B.5 5G RAN and IoT Smart Metering

For the other two scenarios, we use the following subsampling methods: periodical sampling and threshold sampling.

B.5.1 Periodical Sampling

Considering the sophisticated sensing matrix or sensing neural network may not even executable on general network equipment such as a programmable switch or IoT sensor, we only compress the data with subsampling, which is most simple and general compression method. To strictly limit the available communication bandwidth, we evaluate different methods with periodical sampling (or evenly sampling). The reason we do not using random sampling method is it has significant fluctuation sampling rate if look at small windows. In Figure B.8 we show the fluctuation of actual window-wise sampling rate when using random sampling, for instance, if the window size is 50 time stamps, then the peak throughput is 61% higher than average value, whereas the minimal rate will be 66% of the average. The big fluctuation on small window makes the real-time recovery less reliable. **Hence we only use the evenly sampling to demonstrate the different performance of data recovery unless there is adaptive function in corresponding method.**

B.5.2 Threshold-based Sampling

When doing subsampling, the percentile of threshold is set to 97%, which captures top $\sim 3\%$ of the samples. Here the goal is to use the threshold based sampling to avoid missing extreme value, which is very helpful when anomalous sample has significant sparsity. For rest of information we rely on generative model instead of data sparsity, and the 3% samples is small enough to guarantee significant efficiency.

On the 5G RAN and IoT datasets, we did not see any gain of use lower percentile (more samples by threshold). For fully sparsity measurement where only few significant samples are needed, we can simply adapt the sampling rate by changing percentile

instead of sampling rate, but this is leaved for future work as the 5G RAN and IoT scenario requires fine recovery of whole time series. For the ability and robustness in terms of learning the distribution and correct threshold, we evaluate it on both of spectral and temporal domain in §5.6 with metric NWD and NSWd, where the learned distribution is very close the ground truth, and this is sufficient to filter out significant values if needed. Besides, during evaluation we mainly focus on the recovery ability of different methods, and we provide the same threshold based sampling result as DISTILGAN to all baselines.

B.6 Details of Baselines

Sampling

FT-IFT [185] takes a sampled measurement KPI as input and obtains the corresponding frequency domain representation with DFT (Discrete Fourier Transform) at the sender, then at the receiver side uses IDFT (Inverse Discrete Fourier Transform) to reconstruct the data stream (time series). Overall, if the sampling rate is lower than the Nyquist rate then this method is equivalent to passing the sampled data through a low pass filter – no features beyond the half sampling rate (frequency) will be preserved.

Sketching

For the comparison with the sketching methods in the literature, we mainly consider the following works in recent years: **SketchVisor** [69], **SeqSketch** and **EmbedSketch** [71], as they additionally use compressive sensing for recovery process and in general has better efficiency and robustness. **ElasticSketch** [182] and **UnivMon** [101] are considered in some positions in this paper to show the properties of representative state of the art sketching only methods.

Sensing & Recovery

Classical CS. This is the CS algorithm used in various network measurement task [33, 69, 71] (with task specific modification sometimes [33, 71]). Here we adapt the original form because the test dataset does not always have enough dimension to carry out matrix decomposition, which may not always be possible to execute on network devices.

CS-GAN [177]. Here we will focus on using the generator of the CS-GAN. Although in [177], a method to train a paired encoder on sender is provided as well, due to the limitation to run a DNN on network equipment, we would not consider the learned sensing part.

AUDIOUNET [91]. AUDIOUNET models the time series reconstruction as a transform problem between two vectors with same length, and adding high frequency features to the input. AUDIOUNET is able to keep the real samples because of the residual connection between input and last layer, hence we classify it as a deep imputation model without GAN. To apply AUDIOUNET to the task in this paper, the input time series should have the same length, which we achieve via nearest interpolation.

CSDI [162]. CSDI is a state-of-the-art time series imputation method based on deep diffusion, outperforming several other prior imputation models such as [52]. Also compared with other recent deep imputation model such as [102], CSDI has better performance on random missing and more suitable for network measurement task. Hence we select CSDI to represent deep generative imputation methods. As the authors of CSDI showed in their extended version [163], this method (as well as its baselines) is ineffective with high percentage of missing samples (e.g., 90% missing samples); we observe the same phenomenon, as shown in §B.6.1.

TIMEGAN [188] is a representative deep generative model for synthesizing time series data, which has inspired recent time-series data generation methods in the networking area [98, 179]. While the original TIMEGAN is based on taking random noise as input, we adapt it to a conditional time-series generation form that takes sampled measurement KPI time series as input and seeks to recover the original KPI time series.

Short-Time Fourier Transform GAN (STFTGAN) [49] is a spectral domain counterpart to TIMEGAN in that it directly extends the spectrum with adversarial training as opposed to TIMEGAN’s approach to transformation in the temporal domain. STFTGAN itself can be taken as an extension of the previous work on TFNET with adversarial training added. The technique of spectrum extension in STFTGAN has been shown to be effective in recent network traffic generation works such as [179].

B.6.1 Evaluation on synthetic time series

We conduct experiments on synthetic datasets consisting fundamental signals in signal processing. These experiments allow us to visually inspect the behaviour of DISTILGAN in comparison to the baselines we consider.

Synthetic Datasets

For synthetic data, we consider the following signal or stochastic process. By default the model will be trained with 10^5 time stamps and evaluate on 10^6 continuous samples from the same signal or process, so that the amount of training set and evaluation set is long enough to give consistent evaluation of model performance and broadly match our real system scenarios.

- **Sine Waves.** Sine wave can well represent most of simple signal in the nature such as audio signal. Moreover, by mixing multiple sine wave with different phase and frequency, we can easily identify the frequency components in the spectrum. A successful method should recover the frequency components even out of Nyquist Sampling Rate.
- **fractional Gaussian noise (fGn) [17].** Known as fractional Brownian motion (fBm), fGn is a continuous-time Gaussian process, with controllable self-similarity during simulation [39]. High self-similarity (Hurst exponents higher than 0.5) represents highly predictable stochastic processes, whereas low self-similarity cases (Hurst Exponents smaller than 0.5) are closer to Gaussian noise. We evaluate DISTILGAN on different self-similarity fGn simulation data generated with the method in [39].

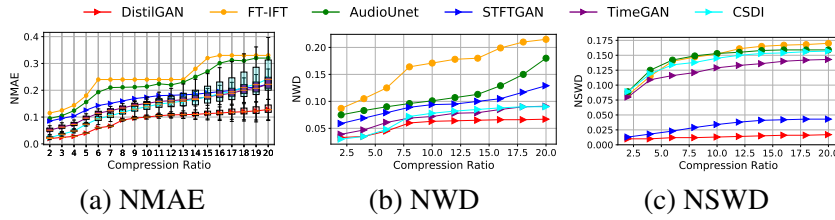


Figure B.9: Performance of DISTILGAN and baselines on beyond Nyquist sampling with sine-waves

Evaluation on sine waves

For sine wave case we consider two scenario: (1) Sampling rate is higher than the Nyquist rate; (2) Sampling rate is significantly lower than some main frequency components (much lower than Nyquist rate).

Within Nyquist Rate. When the sampling rate is within the Nyquist rate, both DISTILGAN and FT-IFT can perfectly reconstruct the signal. For FT-IFT the perfect result is aligned with the Nyquist theorem. The latest generative model such as DISTILGAN and CSDI can almost mimic Fourier transform perfectly, with marginal error ($\text{NMAE} < 10^{-4}$).

Beyond Nyquist Rate. Figure. B.9a shows the NMAE performance of Beyond Nyquist Sampling on synthetic sine waves. For NMAE we show the boxplot because it varies between each inference and we should take the average value if it varies too much, whereas WD based metric is very stable. As the CR goes up (sampling rate goes down), all methods tend to show performance. While the performance of generative models are better than other methods and have similar high fidelity when the CR is lower than 4, the gain of DISTILGAN increases with CR. Also we observe that the CSDI shows too strong stochastic variation when the CR is very high, and some extreme samples can be worse than FT-IFT, though the performance still makes sense on average. Both TIMEGAN and DISTILGAN show better stochastic variation over CSDI when CR is higher than 10 (illustrated in §B.6.4 Figure B.12, where the CR is 16 — $16\times$ downsampling). The quantitative results about stochastic variation and spectrum is illustrated in Figure B.9b and B.9c. All generative models have lower NWD, which means the generated time series has closer distribution to the ground truth, and DISTILGAN shows gain over the other generative model as well. DISTILGAN is also the only method that can recover the spectrum with very high fidelity, much better than the STFTGAN that includes spectrum explicitly.

Evaluation on fGn Series

We also evaluate our method on fGn. When the fGn has a Hurst exponents significantly lower than 0.5, the fGn is more about a Gaussian Noise and unpredictable. Here we evaluate the methods with $H = 0.3$ synthetic fGn. As Figure. B.10a shows, to achieve same NMAE, DISTILGAN can achieve 2.5 to 4 times higher CR than baselines in fGn reconstruction. Also, in Figure. B.10b DISTILGAN shows 50% to 70% lower NWD and 75% to 90% lower NSW than baselines for $15\times$ downsampling. The improvement is also visually significant in Figure B.10c and Figure B.10d, where IDFT based reconstruction lost all high frequency details.

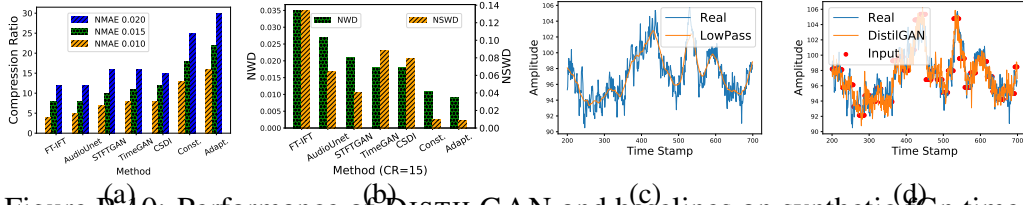


Figure B.10: Performance of DISTILGAN and baselines on synthetic fGn time series in terms of (a) NMAE and CR, (b) NWD and NSWD, (c) FT-IFT for fGn time series reconstruction ($H = 0.3$); (d) DISTILGAN for fGn time series reconstruction.

Takeaways

we get some takeaways from synthetic data evaluation: Real-world time series generally consists of sine waves (may carry information) and random pattern (*e.g.*, noise and anomaly), hence it is reasonable to expect DISTILGAN achieve good performance on different networking measurement time series as well, and we will prove this quantitatively in the rest of this section.

B.6.2 ISP Network Scenario: KPI Breakdown during Adaptive Sampling

Bit-rate KPI reflects more bursty pattern and contributes most in terms of Q-value when adaptive sample on all three attributes together in CAIDA traffic trace. Figure B.11 shows the contribution of different KPIs to Q-Value during flow scale adaptive sampling on CAIDA traffic trace.

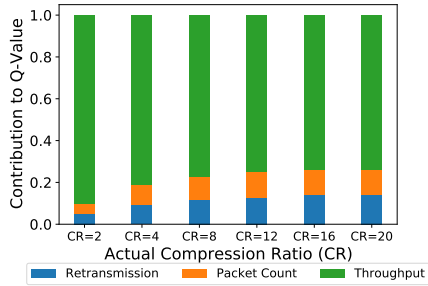


Figure B.11: Contribution to Q-Value During Packet level adaptive sampling, average of all flows. waves (8 times lower than Nyquist rate).

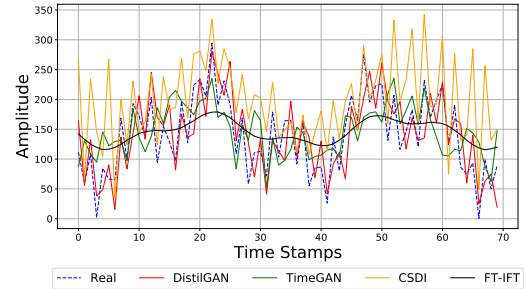


Figure B.12: Beyond Nyquist Sampling and Reconstruction with DISTILGAN, TIMEGAN, CSDI, and the 'HotNets' low-pass filter method in Value During Packet level adaptive sampling, average of all flows. waves (8 times lower than Nyquist rate).

B.6.3 Spectrum Change During DISTILGAN Processing

B.6.4 Synthetic Data Evaluation and Visualization

Beyond Nyquist Sine Wave Reconstruction

In Figure B.12 we observe that the FT-IFT solution lost all high frequency details — though average power is not changed, local power (*e.g.*, between 20 and 25 time

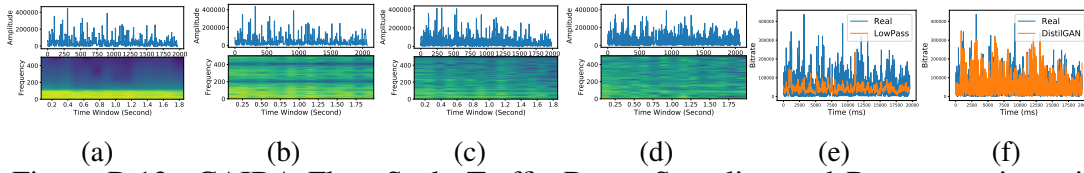


Figure B.13: CAIDA Flow Scale Traffic Down Sampling and Reconstruction with DISTILGAN. (a) spectrum and amplitude after downsampling; (b) after interpolation and spectrum replication. (c) generated spectrum by DISTILGAN. (d) ground truth spectrum and amplitude. (e) FT-IFT for flow traffic (bitrate) reconstruction (amplitude) comparison; (f) DISTILGAN for flow traffic reconstruction.

stamps) changed a lot. TIMEGAN shows better high frequency detail than FT-IFT but does not stay close to the real time series.

CAIDA Data Visualizations

We present the reconstruction result with FT-IFT and DISTILGAN on CAIDA flow bitrate time series reconstruction in Figure B.13e and Figure B.13f, where $CR = 15$ and total length is 20 seconds.

Bibliography

- [1] Chatgpt. <https://chat.openai.com/>. Accessed: 2023-09-01.
- [2] D-ITG homepage. <http://traffic.comics.unina.it/software/ITG/>. Accessed: 2021-10-19.
- [3] General Data Protection Regulation (GDPR). <https://gdpr-info.eu/>.
- [4] Taylor series. https://en.wikipedia.org/wiki/Taylor_series. Accessed: 2023-11-30.
- [5] How to build robust anomaly detectors with machine learning. <https://www.ericsson.com/en/blog/2020/4/anomaly-detection-with-machine-learning>, Apr 2020.
- [6] 3rd Generation Partnership Project (3GPP). Radio measurement collection for Minimization of Drive Tests (MDT). Technical report, 3GPP, <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2602>, 2017. TS37.320-v14.
- [7] 3rd Generation Partnership Project (3GPP). Study on user equipment (ue) power saving in nr. <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3502>, 2019.
- [8] Accuver. XCAL. <https://www.accuver.com/sub/products/view.php?idx=6&ckattempt=1>, 2022.
- [9] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.
- [10] Emmanouil Alimpertis, Athina Markopoulou, Carter Butts, and Konstantinos Psounis. City-wide signal strength maps: Prediction with random forests. In *The World Wide Web Conference, WWW '19*, page 2536–2542, New York, NY, USA, 2019. Association for Computing Machinery.
- [11] Anonymous. List of spotlight KPIs. <https://anonymous.4open.science/r/Spotlight-8E7F>.

- [12] Atlas. Copernicus Urban Atlas. <https://land.copernicus.eu/local/urban-atlas/urban-atlas-2012>, 2012.
- [13] Sanchez B, Ling L, Reif E, Pearce A, and Wiltchko Alexander B. A Gentle Introduction to Graph Neural Networks. <https://distill.pub/2021/gnn-intro/>, 2021.
- [14] Ed. B. Claise. Cisco systems netflow services export version 9. <https://www.ietf.org/rfc/rfc3954.txt>, 2004.
- [15] Richard G. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, 2007.
- [16] Paul Beaumont, Ben Horsburgh, Philip Pilgerstorfer, Angel Droth, Richard Oentaryo, Steven Ler, Hiep Nguyen, Gabriel Azevedo Ferreira, Zain Patel, and Wesley Leong. CausalNex. <https://github.com/quantumblacklabs/causalnex>, October 2021.
- [17] J Beran. Statistics for long-memory processes chapman & hall. *New York*, 1994.
- [18] Davis Blalock, Samuel Madden, and John Guttag. Sprintz: Time series compression for the Internet of Things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–23, 2018.
- [19] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.*, 54(3), apr 2021.
- [20] Leonardo Bonati, Michele Polese, Salvatore D’Oro, Stefano Basagni, and Tommaso Melodia. OpenRAN gym: An open toolbox for data collection and experimentation with ai in O-RAN. *arXiv preprint arXiv:2202.10318*, 2022.
- [21] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9428–9433, 2019.
- [22] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.
- [23] Rhys Alistair Bowden, Matthew Roughan, and Nigel Bean. Network link tomography and compressive sensing. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 159–160, 2011.
- [24] Bouziane Brik, Hatim Chergui, Lanfranco Zanzi, Francesco Devoti, Adlen Ksentini, Muhammad Shuaib Siddiqui, Xavier Costa-Pérez, and Christos Verikoukis. A Survey on Explainable AI for 6G O-RAN: Architecture, Use Cases, Challenges and Research Directions. *arXiv preprint arXiv:2307.00319*, 2023.

- [25] CAIDA. The caida ucsd anonymized internet traces. https://www.caida.org/catalog/datasets/passive_dataset, 2022.
- [26] Jue Cao, Di Kong, Michael Charitos, Denys Berkovskyy, Angelos A. Goulianos, Tom Mizutani, Fai Tila, Geoffrey Hilton, Angela Doufexi, and Andrew Nix. Design and verification of a virtual drive test methodology for vehicular lte-a applications. *IEEE Transactions on Vehicular Technology*, 67(5):3791–3799, 2018.
- [27] Carmelo Cassisi, Placido Montalto, Marco Aliotta, Andrea Cannata, and Alfredo Pulvirenti. Similarity measures and dimensionality reduction techniques for time series data mining. *Advances in data mining knowledge discovery and applications'(InTech, Rijeka, Croatia, 2012,, 1:71–96, 2012.*
- [28] Cellmapper. Cellmapper. <https://www.cellmapper.net/>, 2021.
- [29] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), Jul 2009.
- [30] Kwangrok Chang and Ragil Putro Wicaksono. Estimation of network load and downlink throughput using rf scanner data for lte networks. In *2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8, 2017.
- [31] Ashima Chawla, Paul Jacob, Saman Feghhi, Devashish Rughwani, Sven van der Meer, and Sheila Fallon. Interpretable unsupervised anomaly detection for RAN cell trace analysis. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2020.
- [32] Yang Chen, Dustin J Kempton, Azim Ahmadzadeh, and Rafal A Angryk. Towards synthetic multivariate time series generation for flare forecasting. In *International Conference on Artificial Intelligence and Soft Computing*, pages 296–307. Springer, 2021.
- [33] Yi-Chao Chen, Lili Qiu, Yin Zhang, Guangtao Xue, and Zhenxian Hu. Robust network compressive sensing. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, MobiCom '14*, page 545–556, New York, NY, USA, 2014. Association for Computing Machinery.
- [34] Mayukh Roy Chowdhury, Sharda Tripathi, and Swades De. Adaptive multivariate data compression in smart metering internet of things. *IEEE Transactions on Industrial Informatics*, 17(2):1287–1297, 2020.
- [35] Graham Cormode. Data sketching. *Commun. ACM*, 60(9):48–55, Aug 2017.
- [36] Graham Cormode and S Muthukrishnan. Towards an algorithmic theory of compressed sensing. *Center for Discrete Math. and Comp. Sci.(DIMACS), Tech. Rep. TR*, 25:2005, 2005.
- [37] Christopher Cox. *An Introduction to LTE*, chapter 17. Wiley, 2 edition, 2014.

- [38] Christopher Cox. *An Introduction to LTE*. Wiley, 2 edition, 2014.
- [39] Peter F Craigmile. Simulating a class of stationary gaussian processes using the davies–harte algorithm, with application to long memory processes. *Journal of Time Series Analysis*, 24(5):505–511, 2003.
- [40] Mark E Crovella and Azer Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 160–169, 1996.
- [41] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.
- [42] George Deodatis and M Shinozuka. Auto-regressive model for nonstationary stochastic processes. *Journal of engineering mechanics*, 114(11):1995–2012, 1988.
- [43] Paolo Di Francesco, Francesco Malandrino, and Luiz A DaSilva. Assembling and using a cellular dataset for mobile network analysis and planning. *IEEE Transactions on Big Data*, 4(4):614–620, 2017.
- [44] Emilien Dupont. Learning disentangled joint continuous and discrete representations. *Advances in neural information processing systems*, 31, 2018.
- [45] eBPF. Dynamically program the kernel for efficient networking, observability, tracing, and security. <https://ebpf.io/>.
- [46] Frank Eichinger, Pavel Efros, Stamatis Karnouskos, and Klemens Böhm. A time-series compression technique and its application to the smart grid. *The VLDB Journal*, 24(2):193–218, 2015.
- [47] Lukas Eller, Vaclav Raida, Philipp Svoboda, and Markus Rupp. Localizing basestations from end-user timing advance measurements. *IEEE Access*, 10:5533–5544, 2022.
- [48] Ayman Elnashar and Mohamed A El-Saidny. Looking at lte in practice: A performance analysis of the lte system based on field test results. *IEEE Vehicular Technology Magazine*, 8(3):81–92, 2013.
- [49] Sefik Emre Eskimez, Kazuhito Koishida, and Zhiyao Duan. Adversarial training for speech super-resolution. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):347–358, 2019.
- [50] J. Dugan et al. iPerf – The TCP, UDP and SCTP network bandwidth measurement tool. <https://iperf.fr/>, 2022.
- [51] Mah-Rukh Fida et al. Zipweave: Towards efficient and reliable measurement based mobile coverage maps. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

- [52] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pages 1651–1661. PMLR, 2020.
- [53] Simon Foucart and Holger Rauhut. An invitation to compressive sensing. In *A mathematical introduction to compressive sensing*, pages 1–39. Springer, 2013.
- [54] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5g ran analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom ’23, New York, NY, USA, 2023. Association for Computing Machinery.
- [55] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5g ran analytics and control to a new level. In *ACM MobiCom*, October 2023.
- [56] Xenofon Foukas, Bozidar Radunovic, Matthew Balkwill, and Zhihua Lai. Taking 5g ran analytics and control to a new level. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.
- [57] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, Barcelona SPAIN, 2016. Curran Associates, Inc.
- [58] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [59] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020.
- [60] Ian et al. Goodfellow. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [61] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Dzip: Improved general-purpose loss less compression based on novel neural network modeling. In *2021 Data Compression Conference (DCC)*, pages 153–162. IEEE, 2021.
- [62] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.
- [63] Groundhog. CovMo eVDT. <https://www.ghitinc.com/the-future-of-drive-test-after-covid-19-virtual-is-here-to-stay> 2020.

- [64] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.
- [65] Jiajia Guo, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. Convolutional neural network-based multiple-rate compressive sensing for massive mimo csi feedback: Design, simulation, and analysis. *IEEE Transactions on Wireless Communications*, 19(4):2827–2840, 2020.
- [66] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, Long Beach, US, 2017. Curran Associates, Inc.
- [67] Red Hat. Stress testing real-time systems with stress-ng. <https://wiki.ubuntu.com/Kernel/Reference/stress-ng>.
- [68] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *20th International Conference on Pattern Recognition*, pages 2366–2369. IEEE, 2010.
- [69] Qun Huang, Xin Jin, Patrick PC Lee, Runhui Li, Lu Tang, Yi-Chao Chen, and Gong Zhang. Sketchvisor: Robust network measurement for software packet processing. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 113–126, 2017.
- [70] Qun Huang, Patrick PC Lee, and Yungang Bao. Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 576–590, 2018.
- [71] Qun Huang, Siyuan Sheng, Xiang Chen, Yungang Bao, Rui Zhang, Yanwei Xu, and Gong Zhang. Toward nearly-zero-error sketching via compressive sensing. In *NSDI*, pages 1027–1044, 2021.
- [72] HUAWEI. What is a microburst? how to detect a microburst? <https://support.huawei.com/enterprise/en/doc/EDOC1100086962>, 2023.
- [73] et al. Ian Goodfellow. Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.
- [74] Azam Ikram, Sarthak Chakraborty, Subrata Mitra, Shiv Saini, Saurabh Bagchi, and Murat Kocaoglu. Root cause analysis of failures in microservices through causal discovery. *Advances in Neural Information Processing Systems*, 35:31158–31170, 2022.
- [75] Google Inc. Telephony API of Android. <https://developer.android.com/reference/android/provider/Telephony>, 2021.

- [76] Keysight Inc. Nemo Handy by Keysight. <https://www.keysight.com/gb/en/product/NTH00000B/nemo-handy-handheld-measurement-solution.html?rd=1>, 2021.
- [77] Infovista. TEMS. <https://www.fovista.com/tems>, 2022.
- [78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.
- [79] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12*, page 239–252, New York, NY, USA, 2012. Association for Computing Machinery.
- [80] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017.
- [81] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. Automating diagnosis of cellular radio access network problems. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 79–87, 2017.
- [82] Michael Jiang, Milan Nikolic, Stephen Hardy, and Ljiljana Trajkovic. Impact of self-similarity on wireless data network performance. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240)*, volume 2, pages 477–481. IEEE, 2001.
- [83] Carlee Joe-Wong, Sangtae Ha, Soumya Sen, and Mung Chiang. Do Mobile Data Plans Affect Usage? Results from a Pricing Trial with ISP Customers. In *Passive and Active Measurement (PAM) Conference*, 2015.
- [84] Johan Johansson, Wuri A. Hapsari, Sean Kelley, and Gyula Bodog. Minimization of drive tests in 3gpp release 11. *IEEE Communications Magazine*, 50(11):36–43, 2012.
- [85] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.
- [86] Keysight. Nemo Wireless Network Solutions. <https://www.keysight.com/gb/en/products/nemo-wireless-network-solutions.html>, 2022.
- [87] Caner Kilinc et al. JADE: Data-Driven Automated Jammer Detection Framework for Operational Mobile Networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1139–1148. IEEE, 2022.

- [88] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [89] Paulo Valente Klaine, Muhammad Ali Imran, Oluwakayode Onireti, and Richard Demo Souza. A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks. *IEEE Communications Surveys & Tutorials*, 19(4):2392–2431, 2017.
- [90] Linghe Kong, Mingyuan Xia, Xiao-Yang Liu, Min-You Wu, and Xue Liu. Data loss and reconstruction in sensor networks. In *2013 Proceedings IEEE INFOCOM*, pages 1654–1662. IEEE, 2013.
- [91] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. Audio super resolution using neural networks. *arXiv preprint arXiv:1708.00853*, 2017.
- [92] Gene Moo Lee, Huiya Liu, Young Yoon, and Yin Zhang. Improving sketch reconstruction accuracy using linear least squares method. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 24–24, 2005.
- [93] Mark Leznik, Patrick Michalsky, Peter Willis, Benjamin Schanzel, Per-Olov Östberg, and Jörg Domaschka. Multivariate time series synthesis using generative adversarial networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering, ICPE '21*, page 43–50, New York, NY, USA, 2021. Association for Computing Machinery.
- [94] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International conference on artificial neural networks*, pages 703–716. Springer, 2019.
- [95] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. FlowRadar: A better NetFlow for data centers. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, pages 311–324, 2016.
- [96] Fusheng Lin, Hongyu Wang, Guo Chen, Guihua Zhou, Tingting Xu, Dehui Wei, Li Chen, Yuanwei Lu, Andrew Qu, Hua Shao, et al. Fast, scalable and robust centralized routing for data center networks. *IEEE/ACM Transactions on Networking*, 2023.
- [97] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts. Anomaly detection for time series using vae-lstm hybrid model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326. Ieee, 2020.
- [98] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference, IMC '20*, page 464–483, New York, NY, USA, 2020. Association for Computing Machinery.

- [99] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas Anderson. F10: A {Fault-Tolerant} engineered network. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 399–412, 2013.
- [100] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 334–350. 2019.
- [101] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 101–114, 2016.
- [102] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured atate apace models. *Transactions on machine learning research*, pages 1–36, 2023.
- [103] Yi Lu, Andrea Montanari, Balaji Prabhakar, Sarang Dharmapurikar, and Abdul Kabbani. Counter braids: a novel counter architecture for per-flow measurement. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):121–132, 2008.
- [104] Zhilong Lu, Weifeng Lv, Yabin Cao, Zhipu Xie, Hao Peng, and Bowen Du. Lstm variants meet graph neural networks for road speed prediction. *Neuro-computing*, 400:34–45, 2020.
- [105] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [106] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.
- [107] Bo Ma, Bowei Yang, Zitian Zhang, and Jie Zhang. Modelling mobile traffic patterns using generative adversarial neural networks. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7, 2020.
- [108] Alina Machidon and Veljko Pejović. Enabling resource-efficient edge intelligence with compressive sensing-based deep learning. In *Proceedings of the 19th ACM international conference on computing frontiers*, pages 141–149, 2022.
- [109] Yu Mao, Yufei Cui, Tei-Wei Kuo, and Chun Jason Xue. Accelerating general-purpose lossless compression via simple and scalable parameterization. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 3205–3213, 2022.
- [110] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. How Should I Slice My Network?: A Multi-Service Empirical

- Evaluation of Resource Sharing Efficiency. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 191–206, 2018.
- [111] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, Cezary Ziemlicki, and Zbigniew Smoreda. Not all apps are created equal: Analysis of spatiotemporal heterogeneity in nationwide mobile service usage. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 180–186, 2017.
 - [112] Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International conference on machine learning*, pages 4413–4423. PMLR, 2019.
 - [113] Rashid Mijumbi, Abhaya Asthana, Markku Koivunen, Fu Haiyong, and Qin-jun Zhu. Design, implementation, and evaluation of learning algorithms for dynamic real-time network monitoring. *International Journal of Network Management*, 31(4):e2108, 2021.
 - [114] Dimitar Minovski, Christer Åhlund, Karan Mitra, and Per Johansson. Analysis and estimation of video qoe in wireless cellular networks using machine learning. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2019.
 - [115] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
 - [116] Moshe Mishali and Yonina C Eldar. From theory to practice: Sub-nyquist sampling of sparse wideband analog signals. *IEEE Journal of selected topics in signal processing*, 4(2):375–391, 2010.
 - [117] Moshe Mishali, Yonina C Eldar, Oleg Dounaevsky, and Eli Shoshan. Xampling: Analog to digital at sub-nyquist rates. *IET circuits, devices & systems*, 5(1):8–20, 2011.
 - [118] Edgar Costa Molero, Stefano Vissicchio, and Laurent Vanbever. Fast in-network gray failure detection for isps. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 677–692, 2022.
 - [119] Jerome L Myers, Arnold D Well, and Robert F Lorch Jr. *Research design and statistical analysis*. Routledge, New York, 2013.
 - [120] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10*, page 807–814, Madison, WI, USA, 2010. Omnipress.
 - [121] n.d. How do you generate synthetic data? <https://www.statice.ai/post/how-generate-synthetic-data>, 2022.

- [122] n.d. Keysight VDT toolset. <https://www.keysight.com/gb/en/assets/7018-06582/solution-briefs/5992-3870.pdf>, 2022.
- [123] n.d. Open Signal. <https://www.opensignal.com/>, 2022.
- [124] n.d. Spirent Live2Lab. https://www.spirent.com/assets/u/virtual_drive_test, 2022.
- [125] n.d. Tutela. <https://www.tutela.com/>, 2022.
- [126] JM Nichols and F Bucholtz. Beating nyquist with light: a compressively sampled photonic link. *Optics express*, 19(8):7339–7348, 2011.
- [127] Nokia. Nokia and 3 Indonesia develop Zero Drive Test assessment solution to enhance network quality and user experience. <https://www.nokia.com/about-us/news/releases/2020/08/19/nokia-and-3-indonesia-develop-zero-drive-test-assessment/-solution-to-enhance-network-quality-and-user-experience/>, 2020.
- [128] Eduardo Mucelli Rezende Oliveira, Aline Carneiro Viana, Kolar Purushothama Naveen, and Carlos Sarraute. Mobile data traffic modeling: Revealing temporal facets. *Computer Networks*, 112:176–193, 2017.
- [129] opendev. Kernel-modules: IRQ affinity hint fix-ups. <https://opendev.org/starlingx/kernel/commit/7ded00431675bbab05fe254b90efd08eb335f101?style=unified&whitespace=ignore-all>.
- [130] Alan V Oppenheim and Ronald W Schaffer. Digital signal processing(book). *Research supported by the Massachusetts Institute of Technology, Bell Telephone Laboratories, and Guggenheim Foundation. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975. 598 p*, 1975.
- [131] OSM. OpenStreetMap Overpass API. <http://overpass-turbo.eu/>, 2020.
- [132] Ayça Özçelikkale and Haldun M Ozaktas. Beyond nyquist sampling: a cost-based approach. *JOSA A*, 30(4):645–655, 2013.
- [133] Kihong Park, Gitae Kim, and Mark Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of 1996 International Conference on Network Protocols (ICNP-96)*, pages 171–180. IEEE, 1996.
- [134] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das. Understanding traffic dynamics in cellular data networks. In *Proceedings IEEE INFOCOM*, pages 882–890, 2011.
- [135] JET Penny, MI Friswell, and SD Garvey. Detecting aliased frequency components in discrete fourier transforms. *Mechanical systems and signal processing*, 17(2):473–481, 2003.

- [136] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [137] P. Phaal, S. Panchen, and N. McKee. Inmon corporation’s sflow: A method for monitoring traffic in switched and routed networks. <https://datatracker.ietf.org/doc/html/rfc3176>, 2001.
- [138] Caleb Phillips, Douglas Sicker, and Dirk Grunwald. A survey of wireless path loss prediction and coverage mapping methods. *IEEE Communications Surveys & Tutorials*, 15(1):255–270, 2013.
- [139] Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan. Back to the future: Throughput prediction for cellular networks using radio kpis. In *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, HotWireless ’17, page 37–41, New York, NY, USA, 2017. Association for Computing Machinery.
- [140] Darijo Raca, Ahmed H. Zahran, Cormac J. Sreenan, Rakesh K. Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello. Empowering video players in cellular: Throughput prediction from radio network measurements. In *Proceedings of the 10th ACM Multimedia Systems Conference*, MMSys ’19, page 201–212, New York, NY, USA, 2019. Association for Computing Machinery.
- [141] Vaclav Raida, Martin Lerch, Philipp Svoboda, and Markus Rupp. Deriving cell load from rsrq measurements. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–6, 2018.
- [142] Vaclav Raida, Philipp Svoboda, and Markus Rupp. Real world performance of lte downlink in a static dense urban scenario - an open dataset. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.
- [143] Juan M Ramírez, Fernando Díez, Pablo Rojo, Vincenzo Mancuso, and Antonio Fernández-Anta. Explainable machine learning for performance anomaly detection and classification in mobile networks. *Computer Communications*, 200:113–131, 2023.
- [144] RantCell. What is the significance of drive test in telecom and drive test analysis? <https://rantcell.com/RF-drive-test-analysis-significance.html>, 2022.
- [145] Jeff Rasley, Brent Stephens, Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, John Carter, and Rodrigo Fonseca. Planck: Millisecond-scale monitoring and control for commodity networks. *ACM SIGCOMM Computer Communication Review*, 44(4):407–418, 2014.
- [146] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

- [147] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [148] Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.
- [149] Megha Sahu, Snigdha Damle, and Arzad Alam Kherani. End-to-end uplink delay jitter in lte systems. *Wireless Networks*, 27(3):1783–1800, 2021.
- [150] Martin Sauter. *From GSM to LTE-Advanced Pro and 5G*, chapter 4. Wiley, 3 edition, 2017.
- [151] Rohde & Schwarz. Mobile network testing. https://www.rohde-schwarz.com/us/solutions/test-and-measurement/mobile-network-testing/overview/mobile-network-testing_231692.html, 2022.
- [152] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.
- [153] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang. Characterizing geospatial dynamics of application usage in a 3G cellular data network. In *Proceedings of IEEE INFOCOM*, pages 1341–1349, 2012.
- [154] Telecom Knowledge Share. LTE Drive Test Parameters. <https://telecom-knowledge.blogspot.com/2016/09/lte-drive-test-parameters.html>, 2016.
- [155] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.
- [156] Wuzhen Shi, Feng Jiang, Shaohui Liu, and Debin Zhao. Image compressed sensing using convolutional neural network. *IEEE Transactions on Image Processing*, 29:375–388, 2019.
- [157] Xiaofeng Shi, Matthew Osinski, Chen Qian, and Jia Wang. Towards automatic troubleshooting for user-level performance degradation in cellular services. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 716–728, 2022.
- [158] Joel Shodamola, Haneya Qureshi, Usama Masood, and Ali Imran. Towards addressing the spatial sparsity of mdt reports to enable zero touch network automation. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

- [159] Rajkarn Singh et al. Urban vibes and rural charms: Analysis of geographic diversity in mobile service usage at national scale. In *The World Wide Web Conference*, pages 1724–1734, New York, United States, 2019. ACM.
- [160] Benjamin Sliwa, Hendrik Schippers, and Christian Wietfeld. Machine Learning-Enabled Data Rate Prediction for 5G NSA Vehicle-to-Cloud Communications. In *2021 IEEE 4th 5G World Forum (5GWF)*, pages 299–304, 2021.
- [161] Benjamin Sliwa and Christian Wietfeld. Data-driven network simulation for performance analysis of anticipatory vehicular communication systems. *IEEE Access*, 7:172638–172653, 2019.
- [162] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.
- [163] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *arXiv preprint arXiv:2107.03502*, 2021.
- [164] Jakob Thrane, Matteo Artuso, Darko Zibar, and Henrik L. Christiansen. Drive test minimization using deep learning with bayesian approximation. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.
- [165] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32, 2019.
- [166] Olivier Tilmans, Tobias Bühler, Ingmar Poesse, Stefano Vissicchio, and Laurent Vanbever. Stroboscope: Declarative network monitoring on a budget. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 467–482, 2018.
- [167] Olivier Tilmans, Tobias Bühler, Ingmar Poesse, Stefano Vissicchio, and Laurent Vanbever. Stroboscope: Declarative network monitoring on a budget. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 467–482, Renton, WA, April 2018. USENIX Association.
- [168] Catherine Tong, Emma Rocheteau, Petar Veličković, Nicholas Lane, and Pietro Liò. Predicting patient outcomes with graph representation learning. In *International Workshop on Health Intelligence*, pages 281–293, Berlin, Germany, 2021. Springer, Springer.
- [169] Hoang Duy Trinh, Engin Zeydan, Lorenza Giupponi, and Paolo Dini. Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach. *IEEE Access*, 7:152187–152201, 2019.
- [170] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, 15(6):1201–1214, 2022.

- [171] Paul Tune and Darryl Veitch. Sampling vs sketching: An information theoretic comparison. In *2011 Proceedings IEEE INFOCOM*, pages 2105–2113. IEEE, 2011.
- [172] Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 1(1):1–16, 2016.
- [173] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.
- [174] Yu Wang, Jie Yang, Miao Liu, and Guan Gui. Lightamc: Lightweight automatic modulation classification via deep learning and compressive sensing. *IEEE Transactions on Vehicular Technology*, 69(3):3491–3495, 2020.
- [175] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [176] Lina Wu, Danping He, Bo Ai, Jian Wang, Hang Qi, Ke Guan, and Zhangdui Zhong. Artificial neural network based path loss prediction for wireless communication network. *IEEE Access*, 8:199523–199538, 2020.
- [177] Yan Wu, Mihaela Rosca, and Timothy Lillicrap. Deep compressed sensing. In *International Conference on Machine Learning*, pages 6850–6860. PMLR, 2019.
- [178] K. Xu, R. Singh, H. Bilen, M. Fiore, M. K. Marina, and Y. Wang. CartaGenie: Context-Driven Synthesis of City-Scale Mobile Network Traffic Snapshots. In *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 119–129. IEEE Computer Society, Mar 2022.
- [179] Kai Xu et al. Spectragan: Spectrum based generation of city scale spatiotemporal mobile network traffic data. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 243–258, 2021.
- [180] Kai Xu, Rajkarn Singh, Marco Fiore, Mahesh K. Marina, Hakan Bilen, Muhammad Usama, Howard Benn, and Cezary Ziemlicki. SpectraGAN: Spectrum Based Generation of City Scale Spatiotemporal Mobile Network Traffic Data. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT’21)*, page 243–258. ACM, 2021.
- [181] Guang Yang, Simiao Yu, Hao Dong, Greg Slabaugh, Pier Luigi Dragotti, Xujiong Ye, Fangde Liu, Simon Arridge, Jennifer Keegan, Yike Guo, et al. Dagan: deep de-aliasing generative adversarial networks for fast compressed sensing mri reconstruction. *IEEE transactions on medical imaging*, 37(6):1310–1321, 2017.

- [182] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. Elastic sketch: Adaptive and fast network-wide measurements. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 561–575, 2018.
- [183] Wenzhuo Yang, Hung Le, Silvio Savarese, and Steven Hoi. Omnixai: A library for explainable ai.(2022), 2022.
- [184] Yang Yang, Yang Li, Wuxiong Zhang, Fei Qin, Pengcheng Zhu, and Cheng-Xiang Wang. Generative-adversarial-network-based wireless channel modeling: Challenges and opportunities. *IEEE Communications Magazine*, 57(3):22–27, 2019.
- [185] Nofel Yaseen, Behnaz Arzani, Krishna Chintalapudi, Vaishnavi Ranganathan, Felipe Frujeri, Kevin Hsieh, Daniel S. Berger, Vincent Liu, and Srikanth Kandula. Towards a cost vs. quality sweet spot for monitoring networks. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, HotNets ’21, page 38–44, New York, NY, USA, 2021. Association for Computing Machinery.
- [186] Nofel Yaseen, Behnaz Arzani, Krishna Chintalapudi, Vaishnavi Ranganathan, Felipe Vieira Frujeri, Kevin Hsieh, Daniel S. Berger, Vincent Liu, and Srikanth Kandula. Towards a cost vs. quality sweet spot for monitoring networks. *CoRR*, abs/2110.05554, 2021.
- [187] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 458–472, 2022.
- [188] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.
- [189] Minji Yoon, Bryan Hooi, Kijung Shin, and Christos Faloutsos. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 647–657, 2019.
- [190] Minlan Yu. Network telemetry: Towards a top-down approach. *SIGCOMM Comput. Commun. Rev.*, 49(1):11–17, feb 2019.
- [191] Xinyang Yu, Yanqing Peng, Feifei Li, Sheng Wang, Xiaowei Shen, Huijun Mai, and Yue Xie. Two-level data compression using machine learning in time series database. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1333–1344. IEEE, 2020.
- [192] Yannan Yuan, Jiaolong Yang, Ran Duan, I Chih-Lin, and Jinri Huang. Anomaly detection and root cause analysis enabled by artificial intelligence. In *2020 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2020.

- [193] Chaoqun Yue, Ruofan Jin, Kyoungwon Suh, Yanyuan Qin, Bing Wang, and Wei Wei. Linkforecast: Cellular link bandwidth prediction in lte networks. *IEEE Transactions on Mobile Computing*, 17(7):1582–1594, 2018.
- [194] Chaoyun Zhang, Marco Fiore, and Paul Patras. Multi-service mobile traffic forecasting via convolutional long short-term memories. In *2019 IEEE International Symposium on Measurements & Networking (M&N)*, pages 1–6. IEEE, 2019.
- [195] Jian Zhang and Bernard Ghanem. Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1828–1837, 2018.
- [196] Yin Zhang, Matthew Roughan, Walter Willinger, and Lili Qiu. Spatio-temporal compressive sensing and internet traffic matrices. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 267–278, 2009.
- [197] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. Traffic-GAN: Off-Deployment Traffic Estimation with Traffic Generative Adversarial Networks. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1474–1479. IEEE, 2019.
- [198] Yihao Zhao, Ruihai Wu, and Hao Dong. Unpaired image-to-image translation using adversarial consistency loss. In *European Conference on Computer Vision (ECCV)*, pages 800–815. Springer, 2020.
- [199] You Zhou, Youlin Zhang, Chaoyi Ma, Shigang Chen, and Olufemi O Odegbile. Generalized sketch families for network traffic measurement. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3):1–34, 2019.
- [200] ZTE. One Person, One Car, One Terminal, ZTE WNG Automatic Drive Test Solution Dramatically Improves O&M Efficiency. <https://www.mobileworldlive.com/zte-updates-2019-20/one-person-one-car-one-terminal-zte-wng-automatic-drive>, 2019.