Towards Automatic Troubleshooting for User-level Performance Degradation in Cellular Services

Xiaofeng Shi*
xs374u@att.com
AT & T Labs – Research
Bedminster, NJ, USA

Matthew Osinski
mosinski@gmail.com
AT & T Labs – Research
Bedminster, NJ, USA

Chen Qian
cqian12@ucsc.edu
University of California
Santa Cruz, CA, USA

Jia Wang
jiawang@att.com
AT & T Labs – Research
Bedminster, NJ, USA

ABSTRACT

Troubleshooting cellular service issues at the per-UE (User Equipment) level is an essential task for cellular providers. However, diagnosing service issues at per-UE level is costly because it requires advanced expertise and in-depth inspection of massive network log data. This paper presents NeTExp, a generic and comprehensive data-driven approach to automatically troubleshoot cellular service issues reported by customers. NeTExp determines whether the root cause of a user-reported service issue is from the network side or the device side through deep neural networks, which extract complex spatial-temporal feature profiles from massive network log data. The system is trained and validated using an extensive period of network and customer care data from a major cellular service provider in United States. We also present a case study on an external event that caused cellular service issues in 2020 to demonstrate the effectiveness of NeTExp on detecting network issues and identifying network-issue-related root causes at per-UE level.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network manageability**.

KEYWORDS

Cellular Networks, Troubleshooting, Automation

ACM Reference Format:
Xiaofeng Shi, Matthew Osinski, Chen Qian, and Jia Wang. 2022. Towards Automatic Troubleshooting for User-level Performance Degradation in Cellular Services. In *The 28th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3495243.3560535

*Xiaofeng Shi was a phd student at University of California Santa Cruz when major part of this work was done. Chen Qian and Xiaofeng Shi were partially supported by National Science Foundation Grants 1750704, 1932447, and 2114113.

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10…$15.00
https://doi.org/10.1145/3495243.3560535

1 INTRODUCTION

An essential task of cellular carriers is providing reliable and high-performance cellular services for end-device users. In order to guarantee reliability and improve users' experience, the carriers need to resolve service outages or performance degradation issues experienced by customers. In practice, the issues could be attributed to a variety of reasons, such as network outages/maintenance, device provisioning errors, mobile phone hardware/software bugs, and external events. Many automated functions have been deployed in the current operating cellular networks to monitor the network status and proactively detect the on-going or potential network failures (such as outages or anomalies) [2, 11, 15]. Those systems can effectively detect network issues that impact multiple users in the affected area.

Despite the effectiveness of those proactive issue detection system, not all service issues experienced by the individual customers can be properly solved through the proactive systems. There could still be issues that are user device-specific, such as the problems from the specific user equipment, user device provisioning issues, and there is always the possibility of a network impairment going undetected but impacting the quality of the specific user's experience. In addition, even if the network issue has been known by the provider, the provider also needs to respond to customers about those known issues and resolve their concerns. As a complementary method, upon experiencing those cellular service degradation issues, one traditional way for customers to inquire about and resolve an issue is to *actively* contact the customer care services and report the experienced issues. Then the service provider can respond accordingly regarding known network issues, or *reactively* investigate the root causes and help customers resolve the problems as timely as they can.

The customer reported issues are typically resolved in two phases: the *customer interaction* phase and the *ticket resolution* phase. The customer interaction phase is a troubleshooting process where the customer engages directly with a care agent and receives diagnosis and resolution immediately over phone calls or online chats. However, not every customer-reported issue can be resolved in the customer interaction phase. More complicated issues that cannot be resolved during the customer interaction phase will then be sent to tier-2 support teams (e.g., device team, network support team) in the format of customer trouble tickets. In the ticket resolution

phase, the ticket is routed to a tier-2 team based on the initial assessment of the possible root causes of the issue. It is possible that the initial assessment of the root cause of a ticket is not accurate, and the ticket can be routed through multiple teams before it is successfully resolved.

One key metric to measure the effectiveness of the customer care service is the resolution time for customer-reported issues. To reduce the resolution time, it is critical to (i) minimize the time spent on inspecting the problem and identifying the root cause during the live conversation between the customers and the agents, (ii) minimize the number of customer tickets that need to be sent to tier-2 support teams, and (iii) minimize the number of tier-2 teams that a ticket is routed through before it is resolved. Therefore, an automatic system that can timely and explicitly tell the root cause (i.e., whether the problem is from the network or device side) of the user-reported issue at the early stage of the troubleshooting process can significantly help reduce the average end-to-end issue resolution time cost. For example, if we can quickly determine that a reported issue is related to a known root cause, then there is no need to create a ticket for further investigation. If we can determine a reported issue is not related to any known event and is likely to be network related (instead of device related), then the ticket will be routed directly to the network support team for resolution. It is important to note that these decisions need to be made at *per user device level.*

However, existing automatic cellular network troubleshooting methods [4, 10–13, 22, 28, 31] cannot perfectly meet the above demand, because they are designed to detect network failures only in the cell-level scope. Namely, they mainly focus on detecting the network problems that potentially cause the emergence of the service issues in an area, rather than responding to every individual customer's inquiry in a *reactive* manner during a live care contact. The key challenge for the latter cases is that the issues and the experience scenarios of every individual customer are highly diverse due to a large number of personalized factors of the customers and the areas. The convolution and correlation of these factors make the problem even more complicated. To fill this gap, we propose a generic and comprehensive data-driven troubleshooting system called NeTExp (**Ne**twork **T**roubleshooting **Exp**ert) for identifying the root cause of user-reported issues in the online *reactive* troubleshooting phase. NeTExp can automatically answer the key question in the customer interaction phase: whether the root cause of a service issue reported by the customer is a network problem. To answer this question, NeTExp also needs to determine (1) whether there are any network anomalies that impacted the user in the corresponding serving cells, and (2) whether the user-side symptoms correlate with those network anomalies nearby. Designing and implementing such an automatic system is challenging because 1) jointly modeling the cell-level events and user equipment (UE) level events is difficult as it includes complex spatial and temporal context among cells to cells and cells to UEs; 2) there is no sufficient ground truth resolution data, which is expensive to obtain; 3) the unique features of the cells and the individual customers further complicate the problems. In this paper, we address the above challenges by utilizing and customizing advanced machine learning methods that are capable of modeling the complex cell-to-cell and cell-to-UE network state correlations. In addition, we apply

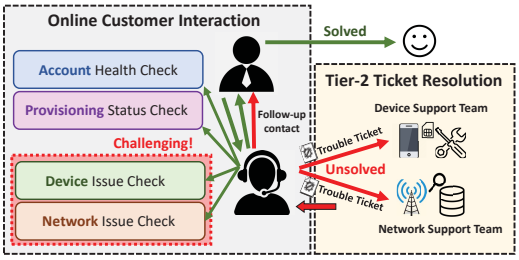


Figure 1: A summary of the troubleshooting process.

the domain knowledge and experiences of running a nationwide cellular network on engineering the features, training the models, and validating the systems. Our contributions are summarized as follows:

(1) We propose a generic framework for automatic service troubleshooting in cellular networks that significantly improves the network problem identification rate and reduces troubleshooting costs.

(2) We make the first attempt to jointly model the complex correlation of the network conditions among the neighboring cell sites and their impacts on the UE of the areas using customized deep learning tools.

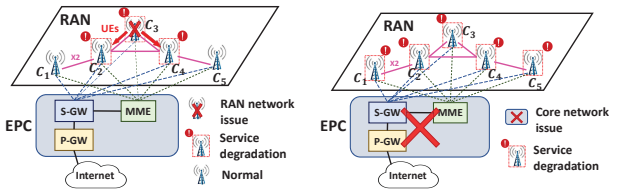
(3) We evaluate the system using massive network log data and care data from a large US cellular provider. We also apply the model to study a historical network problem.

The rest of the paper is organized as follows. Section 2 presents the statement of the problem. The system design is illustrated in Section 3. Section 4 shows the evaluation results. A case study is presented in Section 5. Practical concerns for the system are discussed in Section 6. Section 7 provides the related works.

2 PROBLEM STATEMENT

2.1 Reactive cellular service issues troubleshooting

Upon experiencing cellular service degradation, customers may contact the customer care of the service provider to report and resolve their issues. The reactive troubleshooting and resolution process often consists of two phases: the *customer interaction* phase and the *ticket resolution* phase. A summary of the whole process workflow is illustrated in Fig. 1. During the customer interaction phase, the customer actively speaks to an agent through care calls or online chats. The agent will go through a sequence of designated steps to troubleshoot the service issue while the customer is engaged in the conversation. These troubleshooting steps involve checking customer account status, verifying provisioning status, determining if the customer is impacted by any known events, examining device configuration setting and performing other device-specific diagnoses. While most service issues can be resolved in the customer interaction phase, some service issues may need in-depth investigation before a root cause can be identified. These remaining services issues can be either network- or device-related. The agents will create customer trouble tickets and dispatch them to the Tier-2 support teams for offline inspection. During the customer ticket resolution phase, the inspection often requires gathering and



(A) A RAN issue (B) A core network issue

Figure 2: Example scenarios of network issues.

analyzing measurement data over a time period at both the local network level and individual mobile device level. Depending on the complexity of the issues, the ticket resolution phase usually takes hours to days.

2.2 Challenges in troubleshooting

While some troubleshooting tasks (e.g., checking account and provisioning status) can be executed by software in an automated fashion, troubleshooting network- or device-related issues are largely manual due to the following challenges.

First, troubleshooting a service issue at per user equipment (UE) level is inherently complex. There are a variety of causes of service degradation, including different types of network issues and device issues, many of which produce similar symptoms (such as Internet connection failures, voice call drops, slow data rates, etc.). Therefore, diagnosing based on the UE-side symptom itself is insufficient to identify the root causes. It is particularly challenging to discover the service issues caused by non-fatal or partial network-side or device-side issues. Some of these service issues can be intermittent or chronic. Therefore, precisely determining the root cause of each service issue often requires applying advanced domain knowledge in analyzing a massive volume of network data.

Second, it is not straightforward to discover some network problems on the cell level and estimate the scale of the impacted users and areas. Fig. 2 illustrates two example network issue scenarios in LTE networks. In Fig. 2 (A), the cell site C_3 is experiencing service degradation due to a radio access network (RAN) outage. Consequently, a large portion of UEs that were originally served by C_3 are handed over to its neighboring cell sites C_2 and C_4, which also causes congestion on C_2 and C_4 and impacts the experience of the customers in those areas. In scenario (B), a network issue happens in the core network. The problem may influence the service performance in a wide area. The examples in Fig. 2 (B) show that the impact of a network problem may not only influence the corresponding cells but also propagate to further cells, which makes it challenging to correlate user tickets with some known network issues. In addition, since different network issues present diverse anomaly and propagation patterns, it requires a decent understanding of the event patterns and their correlation among the neighboring cell sites to figure out the impact of a network problem regarding the user-level quality of experience (QoE).

Third, only a small portion of customers report their service issues. Most customers never contact care support upon experiencing a service issue. Depending on the type and severity of service issues, some customers wait for a period of time before they contact customer care. The information provided by customers regarding their service issues can be ambiguous or inaccurate. Due to the high variance of users' behaviors, many issues need extensive investigation efforts.

Dataset	Short Description
Care Contact Log	Logs for the *interaction* phase. Include the time, issue type, resolution, etc.
Trouble Tickets	Handled by the *Tier-2 team*. Include the expert resolutions for hard cases.
Cell-level Network Log	Real-time KPIs of the cell sites. Collected at eNodeB or gNodeB.
UE-level Network Log	Cellular session log for each UE. Includes user ID, time, duration, the accessed cell sites, and session status.

Table 1: Summary of datasets.

2.3 Learning-based troubleshooting

In this paper, we design a learning-based troubleshooting tool that aims at assisting customer care agents during the customer interaction stage. The major objective is recognizing whether the root cause of the issue is from the network side. In addition, the system can also help tier-2 support teams during the ticket resolution stage to identify the possible cell site(s) that caused the service degradation experience on the user side. Thus, the system can significantly reduce the manual investigation involved in the troubleshooting process and hence reduce the overall resolution time.

Table 1 lists the data sources that are widely used or generated during the troubleshooting phases of the state-of-the-practice framework described in section 2.1. The data mainly includes historical customer care contact log and ticket details, and cell/UE-level network statuses such as cell site Key Performance Indicators (KPIs) and user session states. The cell-level KPIs used in this paper include the average number of Radio Resource Control (RRC) connections (which reflects the temporary user population), and the average utilization ratio of the Control Channel Elements (which reflects the congestion status). We design the data-driven automatic troubleshooting system by learning from the above data. In this work, the datasets are obtained from a large cellular service provider in the US. For privacy reasons, all datasets are anonymized to remove any user identifying or personal information.

3 SYSTEM DESIGN

3.1 System overview

We design a learning-based troubleshooting framework NeTExp as shown in Fig 3. NeTExp includes two major modules: (i) a *proactive cell-level* network state prediction model and (ii) a *reactive UE-level* troubleshooting inference model. The proactive cell site level model predicts the likelihood of a cell site to have network issues that impact customers in the covered cells. The UE-level model infers whether a customer-reported service issue is network-related.

During the training phase, the cell-level prediction model is trained using historical usages, user mobility, performance metrics at the cell site level, and customer care contact and ticket data. The UE-level inference model is trained using the output of the cell-level prediction model, the historical UE level usages, user

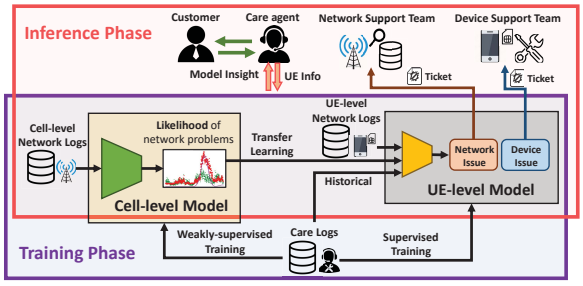


Figure 3: Overview of NeTExp.

mobility, performance metrics, and the customer care contact and ticket data. During the inference phase, the cell-level prediction model proactively predicts the cell sites that are having customer-impacting issues and quantifies the severity of the problem based on the real-time usages and cell site level performance metrics data. Upon receiving a customer contact reporting a service issue, the UE level inference model will take the cell site level prediction on current customer-impacting network issues in the related cells and current UE level usage, mobility, performance metrics to infer whether the customer reported service issues is caused by network-related issues.

Different from prior works, the cell-level model fully considers the interaction among neighboring cell sites, and the UE-level model is the first reactive network issue diagnosis method that is based on the perspectives from both the UE side and serving cell site side, and how the two-side states match each other. This will not only help customer care agents to create a trouble ticket and dispatch it to the corresponding support team for resolution, but also provide network support team enriched information to prioritize and focus on the right cell site for investigation and resolution.

3.2 The cell-level model

3.2.1 Feature modeling. Learning the correlation and interaction between the neighboring cell sites is important for cellular data analysis [25, 26, 35, 39], which is also challenging as it depends on many real-world factors, such as the local distribution of UEs and cell sites, the mobility of the customers, geographic features, and carrier types. To solve this challenge, we design a graph model to represent the interaction between cell sites and propose using the graph convolutional neural network (GCN) [8, 24, 33, 36] to jointly learn the cell site node features and their correlation.

Specifically, the graph model is shown in Fig.4. In the graph G, each node represents a cell site and each edge represents the proximity (weight) between the two neighboring cell sites. The proximity can be defined in multiple ways and is discussed later. On each cell site vertex in the graph, the network condition is represented with a time-series feature acquired by sliding a feature extraction window through the streaming cell-level network log data.

Assume the pair-wise proximity among the k cell sites can be quantified by a 2-D adjacent matrix $A^{k\times k}$ (where each entry $a_{i,j}$ represents the proximity weight from node i to node j), let $G =< V^{k\times m\times w}, A^{k\times k} >$ represent the graph, where V is the $k \times m$ time-series features of the k vertices (i.e., m feature channels for each node, and each channel has time-window length w). Through a GCN layer, the feature on each cell site is recomputed by aggregating the features of itself and the other cell sites in the graph. For example, a typical GCN aggregation rule is defined as:

$$H^{(l+1)} = \sigma((I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}})H^{(l)}W^{(l)}), \quad (1)$$

where $H^{(l)}$ is the node-wise feature input to the layer l ($H^{(0)} = V$), $W^{(l)}$ is a trainable weight matrix that decides how the adjacency matrix $A^{k\times k}$ participates in the aggregation of the features, σ is a non-linear activation function, $I_k - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized graph Laplacian, I_k is an identity matrix, D is the diagonal degree matrix with $D_{ii} = \sum_j A_{i,j}$.

In addition, the temporal feature (i.e., $H^{(l)}$) of the time-series network log data for the cell sites can be encoded by the 1D-CNN layers:

$$h_c^{(l+1)} = \sigma_t(h_c^{(l)} \circ W_t^{(l)}), \quad (2)$$

where $h_c^{(l)}$ is the time-series feature input of one cell site c in the graph ($h_c^{(0)}$ is the raw input feature), $W_t^{(l)}$ is the 1-D temporal CNN kernel, $\circ$ is the 1-D convolution operation, and σ_t is the activation function. Through the two types of the convolution operations, the model is cable of extracting features with complicated spatial-temporal context. The detailed DNN architecture is explained in section 3.2.3.

A natural way to quantify the proximity in the adjacent matrix $A^{k\times k}$ is using the distance among the cell sites [39]. However, we find that distance is not representative enough since the base station selection of mobile devices depends on not only distance but many other factors, such as the geographic features, the density of the cell sites, mobility, etc. Therefore, we propose a new measure for adjacency matrix quantification: **the average number of jointly served UEs by the two cell sites in unit time**. This metric is mainly inspired by the key observation that abnormal state propagation among the cell sites is mainly caused by the hand-offs when a network problem happens to one cell site. Thus, this metric can be a good estimation of how much traffic will be handed off to a neighboring cell sites when network problems happen on one cell site, and is a high-level product of all other unique physical factors in the local area. More importantly, the metric values are easy to obtain by grouping the historical UE-level network log data with time intervals, (anonymous) user IDs, and cell site IDs.

3.2.2 The alternative learning target. NeTExp is required to identify the root causes of the user-reported issues at the per-case level. Therefore, it is ideal to use the manual resolution result of each user's case as the end-to-end learning target. However, it is too costly to identify the whole population of the users impacted by the network issues in practice (including the majority who do not contact the care upon experiencing an issue). Manually selecting and labeling additional cases from the vast user population is rather expensive. Thus, we can only obtain a limited number of ground truth troubleshooting results labeled and verified by expert human agents. The lack of large-scale ground truth data makes it difficult to train a DNN that learns from the high-dimensional data with tremendous spatial and temporal context.

To solve this challenge, we adopt the ideas from weakly-supervised learning and transfer learning [21, 37, 40]. Specifically, NeTExp uses an alternative learning target to pre-train the cell-level model (the

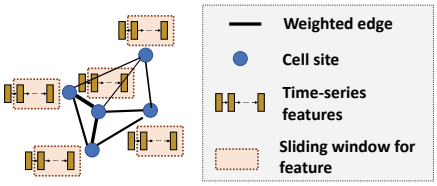


Figure 4: The graph modeling of the cell sites.

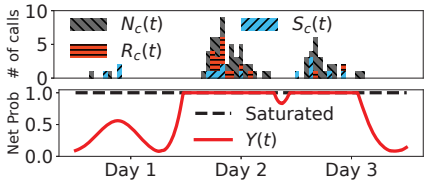


Figure 5: Top: the distribution of the reported service issues. Bottom: The transferred learning target.

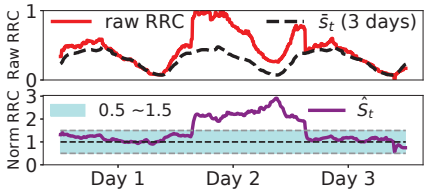


Figure 6: Top: The raw and the expected RRC KPIs (scaled). Bottom: The normalized RRC features.

heavy part of the overall NeTExp system): **how likely the cell is experiencing a UE-impacting network event** at each timestamp t. Although the model with the alternative target cannot directly answer whether a reported issue is a network-side issue, it is expected to provide the high-level representation of the network performance status for the related cell sites, which is an essential insight for case-specific troubleshooting according to experienced troubleshooting operators.

We take the following steps to build the transferred learning target: for each customer, we retrieve the 7-day historical UE network log records before the care contact time, and obtain a set of cell sites that are frequently accessed by the users. Those cell sites are called the "reference" cell sites for this UE. Then we aggregate the total number of customer contacts within a unit time interval by each reference cell site. The aggregation results provide an idea of how many service issues are reported for each reference cell site in each unit time interval. Thus, the intensive gathering of service issues for a reference cell site usually implies network issues in the corresponding cells. Similarly, we measure the aggregation numbers of the service issues that are diagnosed as network issues through the customer interaction phase and the ticket resolution phase using the ground truth troubleshooting tickets of the two phases, which provide extra dimensions about the scale of the influenced users in the area. In this way, we associate the network status observations with the number of tickets received by the customer care services. Unlike existing works that detect network anomalies based on the KPI values, this method is more focused on recognizing the events that impact the QoE of the end-users.

The learning target uses three vectors for each cell site c: $N_c(t)$, the number of total service issues over time; $R_c(t)$, the number of network issues identified during customer interaction; and $S_c(t)$, the number of network issues detected through ticket resolution. If a network issue happened on a cell site (or on its neighbors), a significant increase of N_c, R_c and S_c can usually be observed shortly after the issue occurrence time. An example of such case is shown in the top chart of Fig. 5 (in day 2 and day 3 compared with day 1). Based on this observation, the new learning target, i.e., the likelihood of network issues for the cell site c, can be quantified using N_c, R_c, and S_c. Specifically, we use the 1-D Gaussian Probability Density function $G(t,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{t^2}{2\sigma^2}\right)$ and compute the convolution of density kernel and the measurement vectors over the time dimension: $G(t,\sigma) \circ N_c(t)$, $G(t,\sigma) \circ R_c(t)$ and $G(t,\sigma) \circ S_c(t)$. Then the overall transferred learning target is defined as a weighted sum of the three density vectors:

$$Y(t) = \alpha G(t,\sigma) \circ N_c(t) + \beta G(t,\sigma) \circ R_c(t) + \gamma G(t,\sigma) \circ S_c(t) \quad (3)$$

We normalize $Y(t)$ and let $Y(t)$ saturate at 1 to make the likelihood values in the range $[0, 1]$ and resolve the population-dependent differences. An example of the normalized $Y(t)$ is shown in the bottom chart of Fig. 5. α, β, and γ are decided empirically and should be adjusted based on the effectiveness of the practical troubleshooting phases (N_c, R_c, and S_c) in the wild. Specifically, we look into the known network problems in the history and check the z-scores of N_c, R_c, and S_c during the network issue periods. A larger z-score indicates the corresponding measurement is more important. For example, for our studied cellular provider, we use $\beta \geq \gamma > \alpha$, since the network issue tickets (from both online and offline phases) are more accurate network issue indicators than the total number of care calls.

3.2.3 Model design and training. Fig.7 shows the overall design of the cell-level model to encode the graph-based cell-level features. The neural network is inspired from the STGCN[38] architecture.

The whole cell-level model is used as a feature extractor to learn the cell-level features for each local area. A local area refers to the cells covered by the k neighboring cell sites. In the input feature matrix of height k, the first $m \times w$ feature slice refers to the features of the cell site that directly carries the target UE, while the rest $k-1$ slices are the features of its nearest neighbors ordered by the edge proximity. Once trained, the whole model parameters are consistent for different areas in a large market.

In the input layer, m time-series network KPIs are used as the input features. For the real-number KPI values, we first smooth the data with moving average to denoise the data. Since the traffic loads and capabilities of the cell sites are highly diverse, the KPI data is normalized before being fed for learning. One evident feature for the cell-level KPI data is that the pattern of the KPI time series repeats every 24 hours because of the similar daily traffic patterns. Therefore, we normalize the KPI data by: $\hat{s}_t = \frac{s_t}{\bar{s}_{(t \bmod T)}}$, where s_t is the observed KPI at timestamp t of the global clock, and $\bar{s}_i$ represents the expectation of the KPI of the ith timestamps of a day based on the historical data, T is the number of total timestamps in a day. Thus, the normalized KPI $\hat{s}_t$ represents that at a particular timestamp $(t \bmod T)$ of the day, how the observed KPI compares with the expectation of the KPI for the same time of the day. This normalization method is effective for the KPIs that reflect or are related to traffic loads. For example, Fig. 6, shows the raw and normalized average Radio Resource Control (RRC) connections for the same cell site in Fig. 5. The normalization method makes

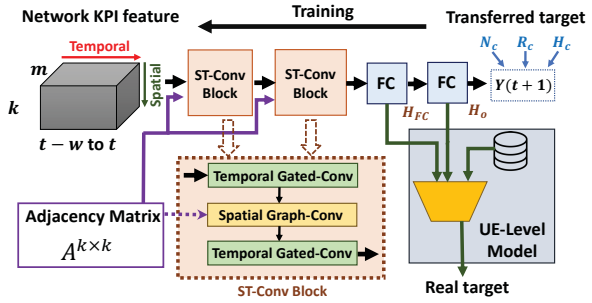


Figure 7: Model design of the cell-Level model.

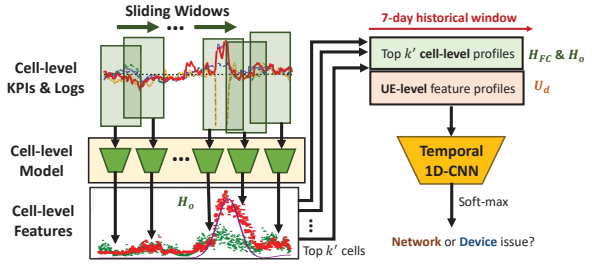


Figure 8: Design of the UE-Level model.

the abnormal network KPIs (in day 2) highly distinguishable. The abnormal RRC KPI states in day 2 of Fig. 6 can well explain the increase of issue reports in day 2 and day 3 of Fig. 5. Thus, this normalization method is an effective outlier detection module to highlight the abnormal KPI values for a given timestamp of the day.

Next to the input layer are the two ST-Conv blocks (as shown in Fig.7) that can encode the spatial-temporal features. In each ST-Conv block, the feature of each cell site is fed into a 1-D temporal CNN layer (Eq.2) with the gated linear units (GLU)[3] as the activation. Then the processed features of all cell sites in the local graph are aggregated using a spatial GCN layer (Eq.1). The GCN block is then followed by another temporal CNN layer for each cell site of the graph to generate the feature representation of the network conditions $H_l^{k\times g\times w_l}$, where g is the number of kernels in the last 1-D CNN layer and w_l is the resampled window size. After the two ST-Conv blocks, the model flattens the feature matrix over the time channel and use a fully connected (FC) layer with kernel size h to compute the $(k \times h)$-D feature representation $H_{FC}^{k\times h}$ of the network conditions on the k cell sites for the sampled timestamp. Thus, $H_{FC}^{k\times h}$ can be used as the extracted feature for the network conditions of the local area at a given time.

In the output layer, the model uses a regression loss function to learn the target Y^k of the k cell sites in the local area. The mean-square-error (MSE) loss is used for training:

$$L(H_o^k(t), Y^k(t+1)) = \frac{1}{k}\sum_i (h_o^i(t) - y^i(t+1))^2 + \lambda L_2, \quad (4)$$

where $H_o^k(t)$ is the output of the model with the input time window that ends at time t, $Y^k(t+1)$ is the transferred learning ground truth of the sampled k cell sites at $t+1$, $h_o^i(t)$ and $y^i(t+1)$ are the ith entry of $H_o^k(t)$ and $Y^k(t+1)$, and λL_2 is the L2 regularization term of the trainable parameters. The model is trained with Adam optimizer [14].

In our implementation, the four Temporal 1-D Gated-Conv layers of the two ST-Conv Blocks have 32, 16, 8, 4 CNN kernels respectively. The size of each kernel is 4, namely, the perceptive field length of the first CNN layer is 20 minutes. The number of neurons h in the feature embedding layer is set as 8. Our validation results show that larger model size provides limited accuracy improvement but more memory cost and overhead. Since our model is executed on CPU servers rather than GPU servers (due to data access restrictions), and the model should learn the network states in real-time for tens of thousands cell sites, we do not choose to use a larger model configuration. The selection of the two key parameters of the input layer, i.e., k and w, is discussed in Section 4.3. After training using the transferred target, NeTExp freezes the parameters of the model. Then it feeds the learned $H_{FC}^{k\times h}$ and H_o^k to the UE-level model as the high-level feature representation of the cell site performance status.

3.3 The UE-level model

The UE-level model is the direct interface for the care agents to learn whether a reported problem is a network-side issue or a device-side issue. Besides the patterns of the UE network logs[20], another important feature is the temporal correlation of the UE-level service errors and the cell-level anomaly states. Thus, we design the UE-level model such that it learns from the features in both aspects.

UE-level features. Based on the historical data session logs for each individual UE, we can create the UE-level feature profiles for learning. Specifically, for each customer who contacts the care, we retrieve the data session logs (time/duration of the session, the accessing cell sites, and the categorical session status) for the target UEs. Then we can create a session usage pattern feature matrix $U_d^{n\times w'}$ for each UE d, where n represents the n-dimension one-hot encoding of the session status, and w' is the historical feature time window size for UE-level trouble inspection. In addition, based on the cell site that the data session is connected with, we use a detailed break-down of session usage features $B_d^{(k'\times n)\times w'}$ for the top k' cell sites that are most frequently accessed by each device d. In our implementation we use $k' = 5$. According to our measurement over a nationwide cellular network for several weeks, the top 5 cell sites contributed to 86% of the cellular sessions and 91% of the usage time on average for each customer. Thus, if a user suffers from a network problem, the cell sites that are responsible are most likely among these top 5 cell sites.

Cell-level features of the reference cell sites. For effective troubleshooting, NeTExp correlates the UE-level profile features with the network status of the top k' reference cell sites. To achieve this goal, NeTExp creates a cell-level profile for the reference cell-sites by using the learned features from the cell-level model, namely, $H_{FC}^{k\times h}$ and H_o^k. For each UE, NeTExp looks back a 1-week historical time window and construct the corresponding feature profiles. The extracted UE-level and cell-level features are concatenated over the *time* dimension for temporal correlation learning. The feature engineering method of the UE-level model is shown in Fig. 8. In the left side of Fig. 8, NeTExp applies the pre-trained cell-level model (Fig. 7) and uses a sliding window to extract the cell-level profile

features over the one week history. The stride of the sliding window is 1 hour.

Modeling and training. The final decision-making model is a CNN classifier that contains several 1-D temporal CNN layers (Eq.2), followed by two fully connected layers with a softmax layer at the end. The learning target is whether the UE's problem is caused by a network-side issue or a device-side issue. Then the UE-level model is trained as a binary classifier using the case-specific manual resolution results from the troubleshooting log data. Since the UE-level model has a much smaller parameter size and only contains a few 1-D CNN layers, it is much easier to train than the cell-level model. Thus, the model can be properly trained using the limited ticket resolution data.

4 EVALUATION

4.1 Datasets

In our experiments, we use nationwide datasets collected from a major US cellular service provider over an extensive period. Specifically, we use the care contact log data (logs for the customer interaction phase) and the trouble ticket data (logs for the ticket resolution phase) as the learning ground truth. In total the dataset includes over 237,000 of care contact records and over 38,000 of customer trouble tickets that reported service problems. All care data instances are used for training via the weakly-supervised learning method and evaluating the cell-level model (Section 4.3). Additionally, we use around 19,000 customer care reports where the root causes of the issues are verified to validate the effectiveness of the end-to-end reactive UE-level troubleshooting (Section 4.4). Moreover, we use the average number of Radio Resource Control (RRC) connections, and the average utilization ratio of the Control Channel Element (CCE) as the cell-level KPI features. The cell-level KPIs are collected for the cell sites that can be associated with the UEs in the customer care datasets. Specifically, for each UE, we first find the top $k^{'} = 5$ cell sites that have most frequently served the UE in the past 7 days before the care call. Then for each serving cell site, we also collect the historical KPI data for its top k neighboring cell sites to build the local cell site node graph. In addition, we extract the UE-level features from the UE cellular session log data. Each log record represents one cellular session, ending with a termination code that indicates why the session is closed. There are 12 distinct codes in our dataset, each of which can be used as a categorical feature that describes the session status. Some example status code include "Normal", "RAT Change" (the radio access technology is changed), "No up/down-link Data Used", etc. In total, more than 325 million UE-level network log records are processed for engineering the UE-level features.

4.2 An example illustration

In Figs. 9, 10, and 11, we use a real example to illustrate the behaviors of NeTExp for handling a specific case. In this example, a customer contacted the customer care and reported service performance degradation. The root cause was not identified after a long period interaction between the customer and the agent.

We then use NeTExp to analyze the root cause of this case. Fig. 9 and Fig. 10 illustrate the RRC KPI and CCE Utilization KPI time series of the relevant cell sites for the past 7 days before the care contact occured. The raw RRC data series are normalized to $[0, 1]$ using min-max normalization for visualization. The "D1" to "D7" on the X axis represent the first to the seventh days of the historical window used for issue inspection. In the top panels of Fig. 9 and Fig. 10, "R1" and "R2" represent the top two cell sites that were most frequently accessed by the user in the past week. From these panels along, we could observe a noticeable increase of connected users (the RRC KPI) and traffic load (the CCE KPI) on R1 during day 6 and 7 compared with the earlier days. The changes are less noticeable on R2. However, those abnormal patterns themselves do not necessarily indicate anomalies on those cell sites, as similar patterns could also be observed in other scenarios such as the gathering of people (e.g., live concerts, sport matches) in the cells.

Next, the cell-level model also investigates the neighboring cell sites of R1. Fig. 9 and Fig. 10 show the KPIs of the top 4 cell sites in R1's neighborhood, i.e., "1st NB" to "4th NB", which are ranked by the graph weights (proximity) to R1 (R2 is the "2nd NB"). The figures show that "1st NB", i.e., the closest cell site, had an outage during day 6 and 7, while the other further neighbors looks normal. Clearly the KPI patterns on R1 were affected by the outages in the neighborhood, although no outage were identified on R1. By using the graph-based model, the cell-level model correctly learns the increasing network issue risk for R1, as shown in the top panel of Fig. 11.

The bottom panel of Fig. 11 shows the session states of the past 7 days for the UE. The rectangles represent the intervals of the cellular sessions of the UE that were carried by each of the two major reference cell sites R1 and R2. Specifically, "Other / Idle" means the device was carried by other cell sites or the device was idle, "R1/R2 Normal" means the sessions with R1/R2 were closed normally, "R1/R2 RAT" means the radio access technology (RAT) was changed. The simultaneous session occupations with R1 and R2 represent that the device was handed off from one to another cell site. By correlating the top and bottom panels of Fig. 11, we can infer that: (1) After the occurrence of the outages (day 6 and 7), the total session length with R2 was significantly reduced and the device was mostly carried by R1. The changes might be due to the network setting updates for resolving the nearby outages. (2) R1 was significantly impacted by the outages in the neighborhood according to the cell level model predictions. From the raw KPI data, we can now infer many other devices nearby were also moved to R1 from their original serving carriers. (3) Along with the carrier changes, the RAT was degraded due to the congestion on R1. Thus, the strong temporal correlation between the cell-level states and UE-level states indicates the root cause of the reported issue was indeed a network problem, which could also be learned by NeTExp using the feature modeling methods in Section 3. In this example, the root network failures were on the neighboring cell sites rather than the major cell sites that served the user. However, as the impact of outages propagated, the actual influenced population size was larger than expected. In fact, the propagation of the network failure impact is usually triggered by the fault tolerance mechanism in current cellular networks. By handing over the customers from the problematic cell site to its neighboring towers, it can dramatically reduce the customer impact of eNodeB/gNodeB failures, although it may cause some congestion on the neighboring cell sites.

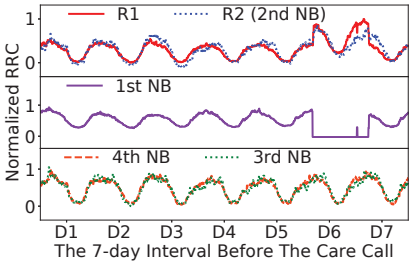


Figure 9: Number of average RRC connections (normalized) - data examples.

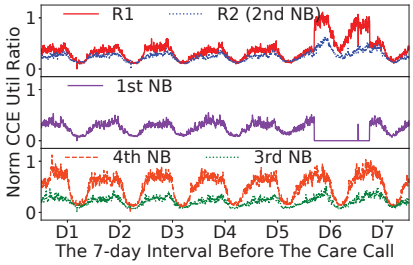


Figure 10: CCE Utilization Ratio (normalized) - data examples.

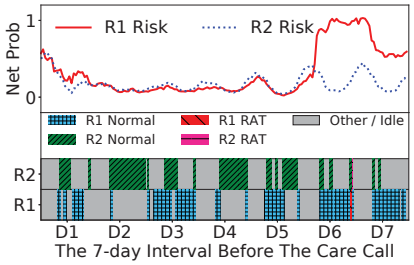


Figure 11: Cell-level outputs (top) and UE-level features (bottom).

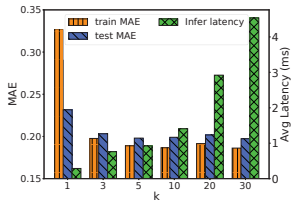


Figure 12: The MAE and cost with different ks.

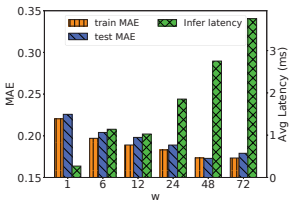


Figure 13: The MAE and cost with different ws.

4.3 Evaluation for the cell-level model

For the cell-level model, we split the cell-level network log data into two parts: we use 2/3 of data for training and 1/3 of data for validation. The mean-absolute-error (MAE) and the model time costs are used as the metrics. Then we analyze the model from both the spatial and temporal dimensions as follows.

Analysis on the spatial dimension. In the spatial dimension, we want to figure out how many neighboring cell sites $(k-1)$ should be considered when predicting the likelihood of network issues in the cells centered by a target cell site. We fix the historical window size w of the KPI data as 12 hours. Then we train the cell-level model with different graph sizes, ranging from $k = 1$ to $k = 30$, where k is the total number of cell sites considered for prediction in the area. For comparison, each model is trained with 10,000 batches and each batch contains 256 samples of the data window samples. The model is trained and executed on a 64-core CPU cluster. The whole training process on the CPU server takes around 2-10 hours for different ks. We randomly select 10,000 cell sites, and measure the mean absolute error (MAE) and the average inference delay (the inference cost for a single cell site at a single timestamp) of the cell-level models. The comparison of different ks is shown in Fig.12. The result shows that the model yields much higher training errors when $k = 1$, namely, it fails to learn the target well when only the target cell site is considered. The results suggest that the interaction of neighboring cell sites is an important feature when analyzing network issues on the cell level. In addition, Fig.12 shows that the model does not improve if more than 5 neighboring cell sites are included in consideration, while the inference latency grows linearly with k. The observation suggests that the transition of the abnormal states indeed exists but only among the nearest neighboring cell sites. The fault tolerance mechanism of cellular networks can dilute the impact of a single network fault on distant cells.

Analysis on the temporal dimension. We next show how the historical window size w impacts the cell-level model. In our evaluation, we set $k = 5$ and change the window size w from 1 hour to 96 hours. We use the similar training and evaluation strategy as discussed above. The comparison result is shown in Fig. 13. The result illustrates that the model accuracy improves when longer time windows are used. The training and validation errors significantly reduce when $w = 48$ hours. As w grows larger than 48 hours, the errors only reduce marginally. Hence, the historical data beyond 48 hours is less important for inferring current network issues. With $w = 48$ and $k = 5$, the model takes less than 3 ms for inferring the network state of one cell site on the CPU server.

4.4 Evaluation of the UE-level model

We evaluate the root cause diagnosing performance of NeTExp using the real historical care contact data introduced above. Specifically, the system is evaluated by 5-fold cross validation and is compared with 5 other baseline diagnosis models. The training of NeTExp for each fold takes around 20 minutes to get converged. A brief introduction of the baselines is as follows:

ICCA (auto) [20]: ICCA is a state-of-the-art cellular issue diagnosis system. It extracts the most discriminative *UE-level* event sequential patterns based on *information gains* by creating a model-based search tree [5] with PrefixSpan [9]. A Gradient Boosting Decision Tree (GBDT) [6] model is then applied for classification. In addition, ICCA [20] also uses manually annotated features which are not available to us. For fair comparison, we only compare the *automatic* feature extraction and learning modules of ICCA.

Fisher Score + KNN: We compute the fisher scores [7] of the cell-level and UE-level features profiled by NeTExp with respect to the root cause categories and select the top n discriminative features for classification. We search the optimal n in range $[5, 1000]$ and select $n = 150$. Then a k-nearest neighbor (KNN) classifier is used for classification.

SRC [32]: SRC (Sparse Representation-based Classification) creates a feature library with profiles of the training data. At inference stage, the model reconstructs the input feature profile of the queried instance through the sparse encoding of the feature library. Then the decision is made by selecting the root cause category that minimizes the reconstruction error with the optimal coefficients.

CNN with only Cell-level or UE-level features: We apply the same CNN classification model introduced in Section 3.3 while using the extracted features from only cell-level or UE-level observations, in order to illustrate the importance of the extracted features from both sides for troubleshooting.

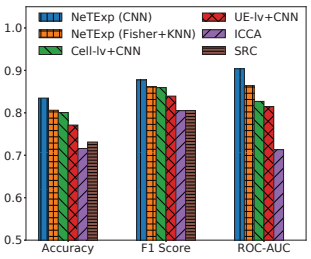


Figure 14: Root cause classification.

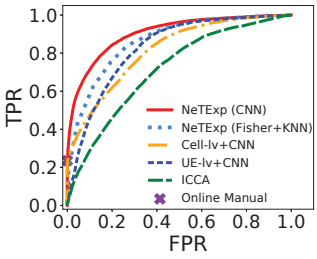


Figure 15: Overall RoC curve.

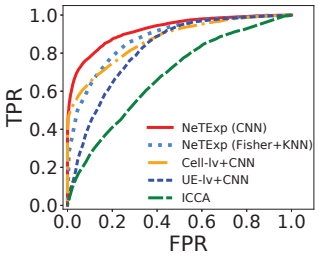


Figure 16: RoC - easy cases.

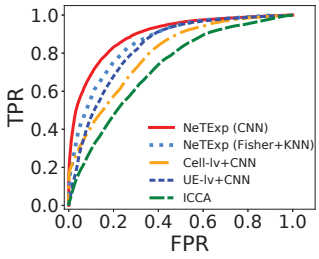


Figure 17: RoC - hard cases.

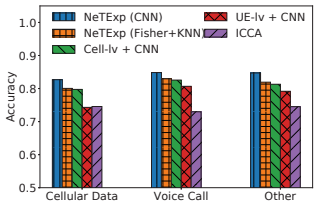


Figure 18: Accuracy for different issue types.

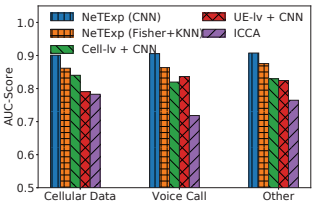


Figure 19: AUC for different issue types.

Learning time costs of the compared models. We used the same CPU cluster to train the compared models. In the (offline) training phase, the major time cost is the feature learning time. Specifically, NeTExp (with $k = 5$ and $w = 48$) needs around 8 hours of training to get a converged cell-level model for network condition feature learning. As a DNN model, NeTExp is expected to be significantly faster for training on GPUs. ICCA (with a 3-hour historical window and 10-layer model-based search trees) needs around 3.5 hours to train the PrefixSpan search tree for feature learning. The time cost for training a binary classifier using the learned features is much smaller. Specifically, the UE-level CNN model in NeTExp takes around 6 minutes, the GBDT model in ICCA takes around 5 minutes, the "Fisher score + KNN" model takes around 6 minutes for training. The inference time cost of all above models is at milli-second level and thus is neglectable for the application use case. The "SRC" model does not need to be trained, while it takes around 0.7 seconds for encoding each instance at inference.

The overall 5-fold validation results (Accuracy, F1-score, RoC-AUC) are shown in Fig. 14. The result shows that NeTExp outperforms other baseline methods for different classification metrics. In addition, as a much simpler and distance-based model, Fisher Score + KNN with the features learned by NeTExp also yields good classification results. This shows that the feature engineering methods in NeTExp provide discriminative features for root cause classification. Fig. 15 presents the ROC curves of the compared methods, where the network-side issue is the "positive" class. The "cross" mark in the figure represents the ratio of network problems that could be manually identified during the online troubleshooting phase. Since many non-outage network issues are difficult to be timely recognized, the recall of the manual network issue identification is low (less than 25%). For the rest 75% cases, the issues are eventually resolved through offline inspection. The result clearly shows that NeTExp significantly improves the recall of the network identification without introducing a large fraction of "false positives". Note that some "false positives" may not be real negatives (i.e., non-network-related issues). This is because that not all network issue cases were successfully identified due to the limitations of the traditional troubleshooting procedures being used in practice. In Section 5, we will show a case study for this type of tickets.

Fig. 16 and 17 present the breakdowns of the performance for different groups of troubleshooting samples. Specifically, we divide the dataset into two subsets based on whether the ticket (if it is network-related) was eventually resolved in the customer interaction stage (the "easier" cases) or in the ticket resolution stage (the "harder" cases). The results show that the features learned from the cell-level data can only work well for identifying the "easier" network issue cases (e.g., a direct network outage), while they fail to work well for the "harder" cases (e.g., indirect issues or chronic issues). On the other hand, the UE-level features can well describe the symptoms on individual user device regardless the types of the network issues. But the symptom on UE-side itself is insufficient for locating the problem, as different problems may produce similar symptoms. Therefore, NeTExp, which correlates both the cell-level and UE-level observations, provides best network issue detection performance. Fig. 18 and Fig. 19 present the accuracy and the ROC-AUC scores of the models for different categories of service problems. The results show that NeTExp outperforms the baseline methods for all different issue categories.

4.5 System efficiency

To understand the feasibility of deploying NeTExp into the existing online customer service framework, we evaluate the end-to-end responding time cost of NeTExp for individual queries. We breakdown the end-to-end time cost of the NeTExp inference module into four parts: **(1) Log query delay**: the cost to read the 7-day historical UE-level logs from database on disk. **(2) UE-feature delay**: The cost to parse the UE-level logs, extract the top k relevant cell site IDs, and create the UE-level feature profiles. **(3) cell-feature delay**: The cost to obtain the cell-level model features for the corresponding cell sites and time windows. **(4) ML-model delay**: The cost for the execution of the UE-level CNN model for inference. We simulate the online query process using 1000 random historical care contacts from a market with more than 2 million users and measure the delay of each components for this market. The average, 90^{th} and 95^{th} percentiles of the time costs of different components are presented in Table 2. The results show that the longest delay is the UE-level network log query delay. The reason is that the UE-level network log data for a million-scale user population market

	Log query	UE-feature	Cell-feature	ML-model
Avg.	11.45	0.073	0.061	0.058
90^{th}	15.29	0.13	0.080	0.063
95^{th}	17.46	0.15	0.086	0.066

Table 2: The average, 90^{th} and 95^{th} percentiles of the time costs (in seconds) of NeTExp components.

is rather massive and cannot by loaded entirely to memory. Thus, the record searching and disk I/O cost for reading the log data of a particular user dominates the end-to-end online process. This part of cost can be further optimized by using faster storage, customized database systems, or a memory cache that prefetches the profiles of the users who are likely to contact the care by forecasting [28]. In addition to the data I/O cost, the feature engineering and model execution costs are neglectable for a live phone call conversation. Taking account of the large I/O cost, the average end-to-end delay is still less than 20 seconds, while a customer care contact usually takes a few minutes or longer. Thus, the inference results generated by NeTExp can well assist the care agent at early stages of the care calls.

5 CASE STUDY

We perform a case study to show how NeTExp can help troubleshoot service degradation issues in practice for a cellular provider. Specifically, we apply our system to analyze an external incident that caused network issues in the U.S. in 2020. During the event period, some areas that were directly affected by the incident experienced network outages, while other areas were indirectly affected due to network changes and maintenance work after the incident.

Fig. 20 shows the distribution of the network-related care contacts in a heatmap and the distribution of the cell sites that are considered to experience network problems (the small blue dots) on Google Map. Specifically, the intensity of the heatmap reflects the aggregated number of care contacts that are thought to be network-related in unit area based on the **ground truth** troubleshooting log data of those few days. Fig. 21 shows the same measures based on the results of **NeTExp**. In Fig. 21, a cell site c is considered to have a network issue during the outage if $\max_t H_o^c(t) > \hat{h}$. The parameter $\hat{h}$ is determined by the analysis of the detection recall, as shown in Fig. 22. Specifically, in Fig. 22, by tuning $\hat{h}$, we show the recall of the cell-level issue detection as a function of the percentage of the detected cell sites over the whole market (i.e., $|C_d|/|C|$, where $|C_d|$ is the number of the cell sites that are considered to have network problems, and $|C|$ is the number of all cell sites). The optimal curve is the green dotted line, which corresponds to the "perfect" detection result based on the ground truth positions labeled in Fig. 20. We select the $\hat{h}$ at the "Threshold" point in Fig. 22, which gives around 75% recall with near to 0 false positive rate. The comparison between Fig.20 and Fig.21 demonstrates that the model accurately learns the impacted locations and how the impact propagated crossing multiple states.

In addition, from Fig.20 and Fig.21, we find that the model-based method tends to recognize more tickets as network-issue related (the positive class). The extra positive instances that are recognized as network problems become the "false positives" using the manual results as the "ground truth" data. However, we find that some "false positives" (FP) are not truly false but due to incomplete/incorrect ground truth data. To better understand the decisions of NeTExp, we show three representative examples from different categories (true positive (TP), FP, and false negative (FN)). In the top charts of Figs. 23, 24, and 25, we visualize a selected representative UE-level feature for the three cases, as well as the 7-day time-series of the learned cell-level model feature $H_o^c(t)$ of their top-1 reference cell site. The bottom charts show the number of care contacts (aggregated for every 3 hours) for the top-5 reference cell sites in the area. The "Outage ts" and the "CC ts" represent the time when the external incident occurred and the time when the customer contacted care.

For customer A, the model infers a network issue, which is a TP case according to the ground truth. The UE-level KPI shows that the device was in abnormal states (the device cannot maintain a stable session) for a period before the care contact time. The temporal consistency of the cell-level and UE-level features demonstrates how the model makes the correct decision. However, the detailed troubleshooting log shows that the customer's issue was not immediately resolved during the customer interaction phase and was eventually resolved by the ticket resolution phase with a longer delay. Thus, NeTExp can help to significantly reduce the delay of a correct response.

Customer B's case shows a "false positive" example. From the trends of the cell-level model predictions and the growth of the care contacts in the local cells, and their close correlation with the UE-level "Credit Drop" failure patterns, we infer that the root cause is indeed a network problem. However, according to the detailed log data, the customer's problem was not resolved during the customer interaction phase, neither was there a troubleshooting ticket generated. Thus, this case study shows a mislabeling case in the ground truth data. In fact, we find this type of case is not unique in our dataset, as the raw care contact log data could be incomplete and noisy due to the difficulty to verify the correctness of the resolutions. Still, NeTExp learns to make correct decisions despite the noise in the training/validation data.

Customer C's case shows a "false negative" decision. The model fails to detect a network issue, although we can manually observe a subtle growing trend of the learned cell-level feature and the number of daily care contacts. We infer the model fail to make a correct prediction because of the inconsistency between the UE-level features and the cell-level features. Specifically, the UE was not able to build data sessions for many days until the past hour before the customer contacted care, which does not match the expected timeline of the outage impact. After inspecting the detailed care contact log, we learn that the customer just activated a new device, which explains why the UE-level network log shows no data session in the earlier days until the hour before the care contact. However, NeTExp does not capture this context from the data, and thus makes an incorrect decision.

Summary of the results. The evaluation results show that NeTExp can more accurately identify the root causes of the individual tickets for different types of network issues and UE-side symptoms. The short end-to-end troubleshooting latency of NeTExp enables early detection of root causes during the customer interaction phases. However, due to the shortages of ground truth and

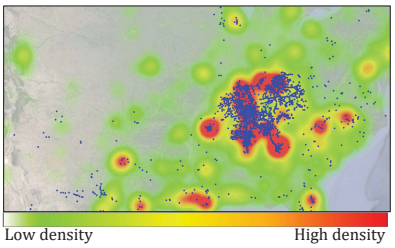


Figure 20: Manual: heat map and issue positions.

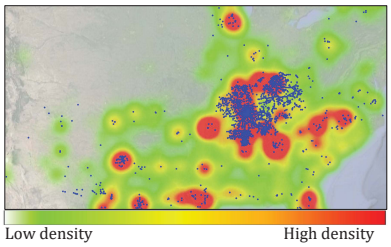


Figure 21: NeTExp: heat map and issue positions.

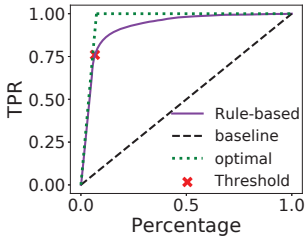


Figure 22: Recall of cell-level issue detection v.s. $|C_d|/|C|$.

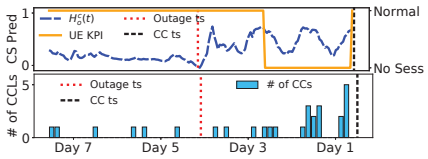


Figure 23: Example case A.

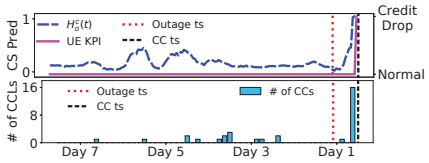


Figure 24: Example case B.

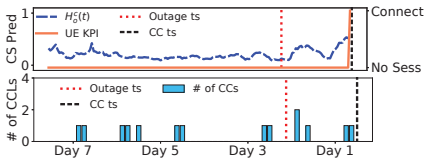


Figure 25: Example case C.

presence of odd corner cases, NeTExp is not a system that can fully replace the manual troubleshooting framework introduced in Section 2.1. Instead, NeTExp should be used as a complementary block in the existing framework to provide reasonable directions and data-driven insights to support the manual troubleshooting process, and thus to help to improve the troubleshooting efficiency and reduce the manual efforts needed to get every customer's query properly addressed.

6 DISCUSSION

6.1 Model updating for unseen scenarios

NeTExp needs to be updated as cellular network configurations and application scenarios change. We summarize the possible network changes and discuss how NeTExp should be updated or retrained.

Traffic pattern changes. The traffic pattern observed in a local cell changes along with the network changes or user behavior changes (e.g., the changes due to the COVID-19 pandemic [34]). Thus, it is necessary to keep monitoring the recent local network KPI patterns and updating the rolling average of the KPIs at each timestamp of the day for effective data normalization. In addition, the model can be periodically fine-tuned using the recent data.

Graph changes. As a generic model that can fit for different subgraphs, NeTExp retraining is not necessary for minor graph changes (e.g., adding/removing cell sites, or updating the proximity weights), whereas the input graph matrix should be updated accordingly. However, if the graph updating is market-wise and dramatic, for example, the cellular provide decides to shut down all 3G services and activate more 5G towers in batch, both the cell-level and UE-level models should be retrained using the new graphs and network measurement data.

New KPI counters. Our current model is trained using the KPI counters available to us, while new other counters or performance metrics might be available in the future. Those new measurements can be easily added as the additional input channels for automatic feature learning using the DNN models, whereas the DNNs need to be retrained.

Model overfitting. We used datasets collected on nationwide to train and evaluate the system in order to mitigate model overfitting. Still, we acknowledge that the model trained using the data from one particular cellular provider may not necessarily work well for another provider given different network scales and use-case scenarios. Thus, the model should be retrained for the change of scenarios.

6.2 Dataset grainularity

The cell-level KPIs used in NeTExp are aggregated and computed in every 5 minutes. Although finer KPI granularity is preferred for fine-grained and accurate network anomaly detection as the data can provide detailed information about jitters, it also requires significantly higher cost to collect, manage, and retrieve the data, and engineer the temporal features with the ML models, especially for a nationwide scale network. Since most users are not sensitive to temporary short network jitters (unless the issue is highly repeated, which can be easily captured) and customers usually do not contact customer care immediately after experiencing a minor issue (for example, a stall in video streaming), it is not necessary to use very fine-grained data.

6.3 Corner cases and model fairness

Corner cases. Unlike manual troubleshooting, the automatic system cannot fully capture the comprehensive real-world context of some complicated corner cases. Besides the corner case example illustrated in the case study (Customer C, i.e., a newly activated device), some other corner case factors could be: unregistrated/mismatched IMEIs (UE IDs), unsupported device models, coverage issues in indoor areas, cloud server issues / congestions on the Internet, intentional interference by nearby attackers, etc. Therefore, if the automatic system fail to make a reasonable diagnosis, the system should roll back to corresponding specialists for manual troubleshooting.

ML fairness. Machine learning based troubleshooting systems may display fairness issue for minor groups or individuals because the model is trained to minimize the loss for the distribtuion of

the entire training dataset rather than individual instances [19]. Despite the fairness issues, the model that yields good accuracy at distribution level can still significantly benefit the customer care workflow by automatically handling most cases and reducing the average end-to-end time/manual effort cost for resolving problems. However, some odd cases still need to be rolled back to or cross-validated by manual troubleshooting. We will further study the fairness problem of ML-based cellular network troubleshooting systems in our future work.

6.4 Fine-grained root cause identification

In this work, the root cause classification is at a coarse level (i.e., whether is issue is from network side or device side). This binary classification result can easily fit into the state-of-the-practice troubleshooting workflow (as illustrated in section 2.1) and currently is a fundamental need by many cellular providers. On the other hand, as a highly representative DNN model, NeTExp learns the high level feature mapping from the raw network observations and generates insightful feature profiles, which provides the potential for fine-grained root cause identification. Unfortunately, due to the shortage of fine-grained troubleshooting ground truth data, it is rather challenging to train and validate the ML model. A possible category of solution is to apply unsupervised or weakly-supervised learning methods integrated with domain knowledge and manual rules. We will continue to explore this problem in our future study.

7 RELATED WORKS

7.1 Data-driven network troubleshooting methods

Along with the tremendous growth of the market for heterogeneous access networks and end devices in the recent decade, manual troubleshooting strategies have become less scalable to the growing device population. Thus, automatic device and network troubleshooting methods that are based on the massive network operation log data or customer care log data have attracted more attention by the service providers and network research community. For example, some data-driven methods use network log and/or trouble ticket data to detect the network anomalies [2, 4, 10–13, 15, 16, 23, 28]. IBM research [4] proposes using a random forest model to predict whether a user will contact the care. CableMon [10] is another learning-based system that can proactively detect network faults in the cable broadband network. Iyer, et al. [11] design a system that is specialized on detecting the RAN issues in the cellular network. All the above existing works target on training a proactive model that can forecast the trend of the emerging network problems or detect anomalies on the network level. However, they cannot directly resolve the user-specific issues in the customer interaction phase. ICCA[20] is the closest work to NeTExp. However, it only uses the UE-level network logs, while the correlation of cell-to-cell and cell-to-UE network states is ignored.

PACE [28] is another automatic cellular service troubleshooting system that focuses on the events of user device level. It solves a binary classification problem: it proactively predicts which users are impacted by a service problem and are likely to contact the customer care as a result of such issues. The focus of this system is prioritizing repair actions for users that are likely to call care. Therefore, this model is trained using data from users whose issues can be resolved using the automated actions that the PACE framework can invoke. On the other hand, our work is primarily used to identify the root cause for users whose issues are non-trivial and therefore need manual debugging. Therefore, the feature engineering and learning methods used by PACE are not ideal for identifying root causes of specific service problems. First, PACE only uses user-level events to describe the symptoms of the service while ignoring service anomalies in the network level which may reflect the root cause. However, different root causes may result in similar symptoms from users' perspective. In addition, the XGBoost model [1] used in PACE is trained based on the ground truth data labeled by the fact whether a user contacted the customer care and triggered a predefined action, which is easy to obtain. Nevertheless, similar model training method can hardly be used for root cause recognition because the ground truth labels are much more expensive to obtain.

Another type of approach focuses on understanding customers' feedback or free-text ticket logs using NLP models [22, 30, 31], and diagnosis based on the described symptoms of devices. For example, NetSieve [22] is proposed to diagnose the problem by understanding the network trouble tickets. LOTUS [31] is a system for identifying which customers are impacted by a local network issue based on the care contact log texts. However, these methods are mainly log-based instead of chatting-based, namely, the problem could only be resolved after a ticket log is generated, rather than during the conversation of the care contacts. This type of methods is orthogonal and complementary to our proposed method.

7.2 ML-based cellular network data analysis

The state of the practice cellular network is getting ever larger and more complex. As a result, there is an increasing demand for automation in cellular network system designs [17, 18, 27, 29, 39]. In recent years, DNNs have become popular for learning cellular network data because of their high capacity of representing spatial-temporal features [27, 29, 39]. For example, DMM [27] uses a recurrent neural network (RNN) model to learn the traveling trace of the UEs based on their cellular data logs. Microscope [39] adopts a 3D Deformable Convolutional Neural Network for mobile service traffic decomposition and network slicing.

8 CONCLUSION

We present NeTExp for automatic and reactive service issue troubleshooting in cellular networks. NeTExp is a generic and comprehensive data-driven approach that considers both the cell site network conditions and UE network logs, as well as their correlations for troubleshooting. The system can be easily extended to incorporate different network KPI data depending on the accessible supporting cellular infrastructures, software, and databases. Our evaluation with an extensive period of real-world data from a major US cellular provider and a real-world case study demonstrates the effectiveness of the system.

REFERENCES

[1] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2015. Xgboost: Extreme Gradient Boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[2] Yi-Chao Chen, Gene Moo Lee, Nick Duffield, Lili Qiu, and Jia Wang. 2013. Event Detection Using Customer Care Calls. In *Proceedings of IEEE INFOCOM*. IEEE, 1690–1698.

[3] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proceedings of ICML*. PMLR, 933–941.

[4] Ernesto Diaz-Aviles, Fabio Pinelli, Karol Lynch, Zubair Nabi, Yiannis Gkoufas, Eric Bouillet, Francesco Calabrese, Eoin Coughlan, Peter Holland, and Jason Salzwedel. 2015. Towards Real-time Customer Experience Prediction for Telecommunication Operators. In *Proceedings of IEEE BigData*. 1063–1072.

[5] Wei Fan, Kun Zhang, Hong Cheng, Jing Gao, Xifeng Yan, Jiawei Han, Philip Yu, and Olivier Verscheure. 2008. Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree. In *Proceedings of ACM SIGKDD*. 230–238.

[6] Jerome H Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of statistics* (2001), 1189–1232.

[7] Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011. Generalized Fisher Score for Feature Selection. In *Proceedings of Uncertainty in Artificial Intelligence*.

[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin* 40, 3 (2017), 52–74.

[9] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. Prefixspan: Mining Sequential Patterns Efficiently by Prefix-projected Pattern Growth. In *Proceedings of ICDE*. IEEE, 215–224.

[10] Jiyao Hu, Zhenyu Zhou, Xiaowei Yang, Jacob Malone, and Jonathan W Williams. 2020. CableMon: Improving the Reliability of Cable Broadband Networks via Proactive Network Maintenance. In *Proceedings of USENIX NSDI*. 619–632.

[11] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. 2017. Automating Diagnosis of Cellular Radio Access Network Problems. In *Proceedings of ACM MobiCom*. 79–87.

[12] Yu Jin, Nick Duffield, Alexandre Gerber, Patrick Haffner, Wen-Ling Hsu, Guy Jacobson, Subhabrata Sen, Shobha Venkataraman, and Zhi-Li Zhang. 2011. Making Sense of Customer Tickets in Cellular Networks. In *Proceedings of IEEE INFOCOM*. 101–105.

[13] Yu Jin, Nick Duffield, Alexandre Gerber, Patrick Haffner, Subhabrata Sen, and Zhi-Li Zhang. 2010. Nevermind, The Problem Is Already Fixed: Proactively Detecting and Troubleshooting Customer DSL Problems. In *Proceedings of ACM CoNEXT*. 1–12.

[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *Proceedings of ICLR* (2015).

[15] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2004. Diagnosing Network-Wide Traffic Anomalies. *Proceedings of ACM SIGCOMM* (2004), 219–230.

[16] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. 2006. Detection and Identification of Network Anomalies Using Sketch Subspaces. In *Proceedings of ACM IMC*. 147–152.

[17] Ajay Mahimkar, Carlos Eduardo de Andrade, Rakesh Sinha, and Giritharan Rana. 2021. A Composition Framework for Change Management. In *Proceedings of ACM SIGCOMM*. 788–806.

[18] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunasish Biswas. 2021. Auric: Using Data-driven Recommendation to Automatically Generate Cellular Configuration. In *Proceedings of ACM SIGCOMM*. 807–820.

[19] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[20] Lujia Pan, Jianfeng Zhang, Patrick PC Lee, Hong Cheng, Cheng He, Caifeng He, and Keli Zhang. 2017. An Intelligent Customer Care Assistant System for Large-Scale Cellular Network Diagnosis. In *Proceedings of ACM SIGKDD*. 1951–1959.

[21] Sinno Jialin Pan and Qiang Yang. 2009. A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[22] Rahul Potharaju, Navendu Jain, and Cristina Nita-Rotaru. 2013. Juggling the JigSaw: Towards Automated Problem Inference from Network Trouble Tickets. In *Proceedings of USENIX NSDI*. 127–141.

[23] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. 2007. Sensitivity of PCA for Traffic Anomaly Detection. In *Proceedings of ACM SIGMETRICS*. 109–120.

[24] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The Graph Neural Network Model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[25] M Zubair Shafiq, Lusheng Ji, Alex X Liu, Jeffrey Pang, and Jia Wang. 2012. Characterizing Geospatial Dynamics of Application Usage in A 3G Cellular Data Network. In *Proceedings of IEEE INFOCOM*. IEEE, 1341–1349.

[26] M Zubair Shafiq, Lusheng Ji, Alex X Liu, and Jia Wang. 2011. Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices. *ACM SIGMETRICS Performance Evaluation Review* 39, 1 (2011), 265–276.

[27] Zhihao Shen, Wan Du, Xi Zhao, and Jianhua Zou. 2020. DMM: Fast Map Matching For Cellular Data. In *Proceedings of ACM MobiCom*. 1–14.

[28] Amit Sheoran, Sonia Fahmy, Matthew Osinski, Chunyi Peng, Bruno Ribeiro, and Jia Wang. 2020. Experience: Towards Automated Customer Issue Resolution in Cellular Networks. In *Proceedings of ACM MobiCom*. 1–13.

[29] Ahmed Shokry, Marwan Torki, and Moustafa Youssef. 2018. DeepLoc: A Ubiquitous Accurate and Low-overhead Outdoor Cellular Localization System. In *Proceedings of ACM SIGSPATIAL*. 339–348.

[30] Pang-Ning Tan, Hannah Blau, Steve Harp, and Robert Goldman. 2000. Textual Data Mining of Service Center Call Records. In *Proceedings of ACM SIGKDD*. 417–423.

[31] Shobha Venkataraman and Jia Wang. 2019. Towards Identifying Impacted Users in Cellular Services. In *Proceedings of ACM SIGKDD*. 3029–3039.

[32] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. 2008. Robust Face Recognition via Sparse Representation. *IEEE transactions on pattern analysis and machine intelligence* 31, 2 (2008), 210–227.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE transactions on neural networks and learning systems* (2020).

[34] Yiting Xia, Ying Zhang, Zhizhen Zhong, Guanqing Yan, Chiun Lin Lim, Satyajeet Singh Ahuja, Soshant Bali, Alexander Nikolaidis, Kimia Ghobadi, and Manya Ghobadi. 2021. A Social Network Under Social Distancing:{Risk-Driven} Backbone Management During {COVID-19} and Beyond. In *Proceedings of USENIX NSDI*. 217–231.

[35] Fengli Xu, Yong Li, Huandong Wang, Pengyu Zhang, and Depeng Jin. 2016. Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment. *IEEE/ACM transactions on networking* 25, 2 (2016), 1147–1161.

[36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful Are Graph Neural Networks? *Proceedings of ICLR*.

[37] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks? *Proceedings of NIPS* (2014).

[38] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *Proceedings of IJCAI* (2018), 3634–3640.

[39] Chaoyun Zhang, Marco Fiore, Cezary Ziemlicki, and Paul Patras. 2020. Microscope: Mobile Service Traffic Decomposition for Network Slicing as A Service. In *Proceedings of ACM MobiCom*. 1–14.

[40] Zhi-Hua Zhou. 2018. A Brief Introduction to Weakly Supervised Learning. *National science review* 5, 1 (2018), 44–53.