Multi-dimensional Impact Detection and Diagnosis in Cellular Networks

Mubashir Adnan Qureshi*, Lili Qiu*, Ajay Mahimkar†, Jian He*, Ghufran Baig*
The University Of Texas at Austin*, AT&T†

***Abstract*—Performance impacts are commonly observed in cellular networks and are induced by several factors, such as software upgrade and configuration changes. The variability in traffic patterns across different granularities can lead to impact cancellation or dilution. As a result, performance impacts are hard to capture if not aggregated over problematic features. Analyzing performance impact across all possible feature combinations is too expensive. On the other hand, the set of features that causes issues is unpredictable due to the highly dynamic and heterogeneous cellular networks. In this paper, we propose a novel algorithm that dynamically explores those network feature combinations that are likely to have problems by using a summary structure *Sketch*. We further design a neural network based algorithm to localize root cause. We achieve high scalability in neural network by leveraging the *Lattice* and *Sketch* structure. We demonstrate the effectiveness of our impact detection and diagnosis through extensive evaluation using data collected from a major tier-1 cellular carrier in US and synthetic traces.**

I. Introduction

Motivation: Cellular networks have been a phenomenal success. The orders of magnitude growth in the number of connected devices and base stations calls for not only innovation in wireless communication but also advances in network management and analytics. Timely detection and diagnosis is required to accurately capture any unexpected impacts and resolve them. Today, network operation teams can detect and diagnose massive service impacts introduced by network changes. Performance impact is a statistically significant change in form of a degradation, or an improvement. An impact is categorized as massive based on its severity and is typically actionable by the operation teams (*e.g.*, bug fixes or reboots to resolve a degradation).

Cellular networks are very heterogeneous due to different locations, parameters, and user behaviors. Each base station in the network is associated with a vector of features (*e.g.*, base station software, hardware, radio parameters, handover parameters, carrier frequencies, and phone types). It is easy to catch impact at the entire network level or an individual base station using statistical analysis. Existing capabilities are well equipped to deal with massive impacts that are observed at coarse granularities (*e.g.*, network-wide impacts) or finer granularities (*e.g.*, individual base stations or edge routers) as relevant alarms are raised. However, if an impact is induced due to a specific feature combination and such impact is not large enough at the network-wide level, the impact can go unnoticed. Motivated by this observation, we explore how to automatically discover the feature combinations that experience significant change and pinpoint root causes.

Challenges: Analyzing performance impact over every feature subset can be very expensive as search space is exponential. Ideally, we want to restrict ourselves to only explore "promising" subsets – subsets which show significant performance impact. The main problem is how to extract these promising subsets. If we start our analysis from coarse granularity, performance impact can be diluted and canceled by other traffic that does not experience major changes or experiences opposite performance impact. Moreover, at the base station, the impact may not be significant enough to raise a statistical alarm.

Localizing the set of features that contribute to the performance impact is also very challenging in such scenarios. This is because the aggregate that is a subset or superset of problematic features can also show significant performance impact. Suppose we have two problematic features A and B. If we aggregate performance impact over data points with feature A or B only, we might see similar performance impact. Moreover, analysis of performance over aggregate A, B, C where C is another feature will also show strong performance impact. In such scenarios, we want to find the set that most accurately characterizes the performance impact (*i.e.*, the intersection of A and B in this case).

Our approach: In this paper, we develop principled approaches for **multi-dimensional impact detection** and **root cause analysis**. We first propose an analysis structure *Lattice-Sketch*, which maintains statistics over feature combinations and on-demand zooms into a fine-grained level if an impact is observed at a higher level. This on-demand zoom-in is enabled using summary algorithm *Sketch*, which reveals the impact of a fine-grained level at a coarse granularity.

After the performance impacts are identified, the operators need to diagnose the set of features that lead to the impacts. We develop an algorithms based on neural network. It takes the performance associated with various feature combinations as the input and output the root cause by learning a function that maps from the feature values to the performance. We choose neural network because it can learn non-linear functions. We further leverage the *Lattice-Sketch* to reduce dimensionality of our root cause analysis.

To demonstrate the effectiveness of our work, we evaluate our approaches using both real and synthetic data. Our results show that using a sketch dimension of 5 allows us to accurately detect anomalies involving multiple features in carrier data. We stress test different parameters in our algorithm to show

how our algorithm scales with varying numbers of features and anomalies. Moreover, we observe high accuracy of root cause detection using our neural network approach.

II. Multi-Dimensional Anomaly Detection

Performance anomalies can be very challenging to detect in cellular networks due to a high heterogeneity, and different scales at which they affect the network. Some anomalies affect only a subset of network and cannot be captured by statistical analysis applied to overall traffic. Current practice used by the operation team is ad-hoc. Some providers just keep track of statistics associated with individual features (*e.g.*, number of dropped calls for certain hardware version), but fail to detect anomalies when they occur due to an interaction between multiple features. One may try to keep track of statistics corresponding to all possible feature combinations, and apply a standard change detection technique to detect if any feature combination experiences an anomaly. For N binary features, there are a total of $\sum_{i=1..N} 2^i * C_i^N = 3^N$ feature combinations, which is prohibitively expensive for a large N. For simplicity and understanding, we use binary features for mathematical analysis. However, as we will show later that we apply our algorithm on carrier data which has non-binary features. As networks densify in 5G, the number and cardinality of features is bound to increase, which exacerbates the scalability issue.

A. Problem Formulation

Input: Input to an anomaly detection engine is a stream of records where each record is defined by a set of feature values and associated performance metric for that record: $(f_1, f_2, ..., f_N|p)$, where f_i is the i-th feature value and p is the associated performance metric. Numerical features can be converted into categorical features using domain knowledge. For example, throughput can be mapped to high/low traffic. Depending on the data, the performance metric can be call dropping probability, latency, throughput, or loss rates etc. Network operators also have historical records, which can be used for comparison. An epoch can be minutes, hours, days or weeks and it depends on the time granularity at which the anomalies need to be detected. So a final record becomes $(f_1, f_2, ..., f_n|p_{-K:-1}, p_0)$, where $p_{-K:-1}$ and p_0 denote the performance over the previous K epochs and the current epoch, respectively. Our goal is to build a detection engine that automatically detects these combinations without exploring the whole search space which is exponential.

B. Lattice based Anomaly Detection

Lattice is used to explore performance for multi-way feature interactions. Lattice starts with a root at level 0, splits into multiple branches based on one feature value at level 1, and splits into more branches based on a combination of two features at level 2, and so on. In general, the nodes at level n capture the statistics for combinations of n features. Building a complete lattice can be expensive for large number of features or features with high cardinality. To improve the scalability, we explore only those branches that see anomalies. Specifically, we start with only the first level of lattice and use the incoming data to update the statistics for the first level only. If a node becomes anomalous (statistics cross a threshold), then we expand our statistics collection to next level. Our goal is to grow the lattice to include all possible anomalies while minimizing the lattice size.

Limitations: The above approach is intuitive, but anomalies involving multiple features may not be big enough to show up at a coarse level. For example, consider there is an anomaly in a 3-way feature combination: $f_1 = 0, f_2 = 0, f_3 = 0$. Aggregation at $f_1 = 0$ dilutes the effect of anomalous traffic if there is a lot of non-anomalous traffic. This does not trigger further lattice expansion and thus fail to detect anomalies.

C. Lattice-Sketch based Anomaly Detection

In order to efficiently catch smaller anomalies, we seek to represent the incoming data in such a way that we can reliably detect the existence of anomalies. There has been works on analyzing massive data streams (*e.g.*, counting the number of flows [1], heavy hitter detection [2], change detection [3]) without keeping lots of state. In these works, a flow is a key value pair (k, v). Since the number of flows can be massive, one can't afford to keep per-flow state. Instead, summary structures are used that can answer queries in an approximate fashion with provable probabilistic accuracy guarantees. These data structures are effective in detecting heavy hitters. But for anomaly detection, we need to detect a large change. Unlike heavy hitters, where an ancestor of a heavy hitter is also a heavy hitter, an ancestor of a node that experiences a large change may not see a large change due to dilution and/or cancellation. Therefore the existing works do not apply in our context. We use *Sketch* to detect changes in performance at higher lattice layers.

Linear sketch: Linear sketch is a streaming algorithm that projects the original data into a much smaller dimension by computing Ax where A is an $n \times D$ matrix, x is incoming data of size D, and $n << D$. A classic linear sketch [4] maintains n counters c_i and n independent hash functions h_i that maps the key to $-1, 1$. When a new data record $(key, value)$ arrives, it updates each counter as follows: $c_i = c_i + h_i(key)value$. Then it estimates the second moment of the data as $c = \sum_i c_i^2/n$. Averaging across the sketch dimensions helps reduce the variance. Then one can determine if there is an anomaly by checking if the second moment estimator is larger than a threshold. The intuition is that non anomalous data are likely cancelled out since their coefficients are on average evenly distributed over $1, -1$. According to the Chebyshev inequality, it can be shown that the classic sketch has false positive and false negative to be $1/k^2$ when we use $|x - \mu| \geq k\sigma$ to detect anomalies where μ and σ are the mean and standard deviation. However, we find the classic sketch works well when there are very few anomalies, but its accuracy degrades significantly when there are more anomalies because the anomalous data may partially cancel each other and reduce variance.

Our sketch: We develop a variant of sketch to detect if an aggregate of flows contains one or more anomalous flows. Instead of mapping $h_i(key)$ to $-1, 1$, we map the key into $[0, 1)$. Anomalies are detected whenever any of the sketch dimensions has c_i larger than a threshold. By using non-negative coefficients, we avoid cancellation in anomalous records. By checking individual sketch dimension, we increase the chance of catching anomalies.

Lattice-sketch: We apply our sketch to lattice. Each lattice node is associated with a multi-dimensional sketch, where each dimension is a random linear combination of performance metrics of the records matching that node. For example, consider a lattice node $f_0 = 0$. Any records that match $f_0 = 0$ will be used to update the lattice node as follows. For the i-th sketch dimension and j-th record, we hash the complete feature vector to a random coefficient ($c_{i,j}$) and increment the corresponding sketch dimension by $c_{i,j}p_j$, where p_j is the performance associated with the incoming record. In this way, the K-dimensional sketch is essentially

$$\sum_{j=1} c_{0,j}p_j, \sum_{j=1} c_{1,j}p_j, ..., \sum_{j=1} c_{K,j}p_j$$

Consider a dataset with 2 binary features (f_0, f_1), it has 4 possible feature vectors: $(f_0 = 0, f_1 = 0), (f_0 = 0, f_1 = 1), (f_0 = 1, f_1 = 0), (f_0 = 1, f_1 = 1)$. For each sketch dimension, we have a random coefficient corresponding to each feature vector. When a new record $(f_0 = 1, f_1 = 1, p = 0.1)$ comes in, we update the sketch value associated with three lattice nodes: $f_0 = 1$, $f_1 = 1$, and $(f_0 = 1, f_1 = 1)$. For each lattice node, we increment each of its sketch dimension k by $c_{k,j}p = c_{k,j} * 0.1$.

Detection rule: An anomaly is detected if any sketch dimension c_i sees a change greater than $\mu_i + 3 * \sigma_i$. Here μ_i and σ_i denote the mean and standard deviation of sketch values for dimension i over the previous epochs. For example, consider 3 binary features. There is an anomaly in $(f_0 = 0, f_1 = 0, f_2 = 0)$, abbreviated as $(0, 0, 0)$. The anomaly only accounts for 10% traffic. If we sum up the performance of all traffic matching $(0, *, *)$ at $f_0 = 0$, the anomaly is hidden since most traffic is normal. In comparison, by using k dimension summary, we generate k random coefficients for each feature combination. If $(0, 0, 0)$ is assigned a large random coefficient in at least one of the k dimensions, the anomaly will show up at a coarse level and trigger in-depth analysis. Moreover, as long as any of the ancestors of $(0, 0, 0)$ (*e.g.*, $(0, *, *)$, $(*, 0, *)$, or $(*, *, 0)$) detects an anomaly, the lattice will be expanded as desired. Therefore, sketch significantly increases the likelihood of catching anomalies involving multiple features.

Exploration strategies: Once a lattice node A sees a large change in at least one sketch dimension, we explore this branch further. For example, when a node $f_0 = 0$ detects a large change in some of its sketch dimensions, it will explore the 2-way feature combinations that involves $f_0 = 0$.

The above condition alone may lead to exploring the lattice unnecessarily deep since children of anomaly nodes tend to also experience anomalies. For example, if the anomaly is $f_0 = 0$, more fine-grained feature combinations involving $f_0 = 0$ (*e.g.*, $f_0 = 0, f_1 = 0$) may also see anomalies.

To address the issue, we propose if a node and its parent see similar performance distributions, we do not expand the branch further even if the anomaly detection threshold is reached at the child. We experiment with different methods to test the similarities between two distributions (*e.g.*, KS-test, Anderson-Darling tests), and find the simple z-statistics based on the mean (μ_p, μ_c) and variance (σ_p, σ_c) works quite well. z-statistics is computed as $\frac{\mu_p - \mu_c}{\sqrt{\sigma_p/n_p + \sigma_c/n_c}}$, where n_p and n_c correspond to the total number of samples at the parent and child, respectively. If it is larger than 2.5, we consider the distributions are different. After adding this check, it is rare for a lattice to grow beyond the level at which the anomaly is present, thereby improving scalability.

If we have additional information about the feature interaction (*e.g.*, from domain knowledge or historical information), we can further enhance the scalability. In particular, instead of blindly expanding all branches underneath the feature A, we can skip exploring the branches that are known to have no interactions with A.

Initialization: When we find a node has an anomaly, we go down one layer deeper in the lattice to explore more fine-grained combinations. Since we did not keep statistics for the new combination until its creation, we need to consider how to initialize them. A simple option is to initialize the new node to 0. But it may take a long time for the new node to converge to the actual value. Instead, we initialize using the parents' values based on the independence assumption, *i.e.*, the dropping probability at the child is the product of the dropping probabilities at its parents. Our evaluation shows that this simple initialization works quite well.

III. Root Cause Localization

Given the anomalies, we seek to determine their root cause. A problematic configuration will induce anomalous behavior in multiple feature combinations. Consider that feature *A* is the primary problem, then whenever *A* appears with another feature say *B*, the combination (*A,B*) will also show an anomaly. One possible approach for root cause localization is to examine the relationship between a node and its children in the lattice. For example, if most children experience anomalies, their parent (or ancestor) is likely to be the root cause. However, this heuristic is not reliable due to measurement noise and probabilistic nature of anomalies. We propose a neural network based approach to learn the model that maps performance at various feature combinations to root causes. Neural network is flexible and can handle non-linear relationships. Therefore, we use it for our root cause localization.

We start with a simple NN model. Its input is the performance corresponding to each feature combinations from 0..0 to 1..1. If the performance of certain feature combinations is unknown, the input is 0. For binary features, the input size is $O(2^N)$. The output is probability that the corresponding

feature combination is the root cause. Based on the above input and output, we generate data by injecting synthetic anomalies to either the real or synthetic data. Refer to Section IV for details about anomaly injection. We use $80\%/20\%$ data split for training/testing. Each sample is an instance containing different datasets with unique root cause feature combinations. During training, we feed the performance across various feature combinations as the input and adapt the neural network to match the corresponding root causes at the output. For training, the output for root cause feature combinations are set to 1 and the remaining feature combinations are set to 0.

Neural network: We build a neural network where each layer is fully connected. Sigmoid activation function is used at the last layer so that the output is a probability between 0 and 1 and use Relu activation at the other layers. We use ADAM as the optimizer to train the network. We define the loss function as the sum of categorical cross-entropy loss and L2 regularization penalty to avoid overfitting. We find 5 layers and 100 iterations give good performance. Further increasing the number of layers and number of iterations does not significantly improve the performance.

# Features in Anomalies	2	3	4	5
Accuracy	0.99	0.99	0.97	0.97

TABLE I
BASIC NN: TOTAL # FEATURES IS 10. # ANOMALIES IS 1.

As shown in Table I, this approach can identify the root cause of anomalies in the data with an accuracy higher than 97% under a wide range of scenarios. While this approach has high accuracy, it does not scale with the number of features, since the input and output growth is exponential with the number of features.

Sketch-based neural network: Encouraged by performance of the above neural network, we seek to address its scalability issue. In particular, the 2^N input size not only consumes significant storage, but also takes a long time to train and test due to a large neural network size. To enhance scalability, instead of using raw records as the input, we use the sketch at the first layer of the lattice as the input. Specifically, as described in Section II, a sketch at each node consists of multiple random linear combinations of all the records that match the node. Let K denote the sketch dimension and N denote the number of features. Without loss of generality, consider binary features. Each lattice has K dimension, and there are $2N$ nodes at the first layer. So the input size is $2NK$ and much smaller than 2^N as K is a small constant.

Intuitively, sketch is essentially a summary of the database by projecting 2^N dimension onto $2NK$ dimension. While there are only $2N$ nodes at the first dimension, the multi-dimensional sketch at these nodes captures the variation across different feature combinations to some degree. How much it captures the original records depends on the sketch dimension K. We can trade off between the compression rate vs. accuracy by adapting K. As we will expect, the accuracy increases with the sketch dimension. For example, as we will show in Section IV, using $K = 5$ allows us to accurately diagnose 1 anomaly, and using $K = 12$ allows us to accurately diagnose 5 simultaneous anomalies. Important lesson that we learn here is that transformation and compression could be used on the input before feeding to neural networks to improve efficiency and sketch could be a useful transformation to consider.

IV. EVALUATION METHODOLOGY

We evaluate our approaches in three ways: (i) real data with real anomalies, (ii) real data with synthetic anomalies, and (iii) synthetic data with synthetic anomalies. (i) reveals the actual anomalies and root causes in the carrier data and demonstrates the utility in real network operation. (ii) helps us to systematically evaluate the impact of different factors (*e.g.*, the number of anomalies and the number of features involved in the anomalies) and also allows us to quantify false negatives, which requires knowledge about all ground truth anomalies and is not available in (i). (iii) further stress tests the scalability of our scheme.

Real Data: We obtain data from a major tier-1 cellular service provider in US. The service provider captures call detail records (CDRs) for voice calls and data sessions. CDRs are collected in real-time at core network switches (MSC, S-GW/P-GW). From this data, we extract #successful/dropped phone calls for each phone type at each base station. We have 57 phone types. These records are indexed by date and time of the day. The configuration snapshot for each base station is logged into an Oracle database on a daily basis. The configuration captures information such as the software version (8), hardware version (11), carrier frequency (6), morphology (3), RX-state (5) and TX-state (4), carrier aggregation (4) and phone type (57). Morphology is whether it is an urban, suburban or rural area. Numbers in () are the number of different values for each feature. In our formulation, the key performance indicator (KPI) is computed for a given epoch, which lasts for a day/week. We use *Retainability* as KPI which is the ratio of dropped calls to total calls. After combining features from CDR data with base station configuration data and calculating KPI values (current and previous K epochs), our record takes the following form, which has a feature value vector and KPI vector, namely feature sets: $f_0, f_1, ..., f_N$ and KPI sets: $KPI_{-K}, KPI_{-1}, ..., KPI_0$.

For evaluation, a stream of records for a week are fed as an input to our algorithm. We use the data collected during October 2018 – March 2019 from 5 markets in different regions of US. We use a total of 8 features as described and their cardinalities vary from 3 to 57. The operators examine these features first when facing any major problem. We have around 20,000 unique combinations of 8 features in our dataset. Since not all configurations are present in our data, the complete lattice has 100,000+ nodes.

Synthetic Data: To stress test our approach, we also generate synthetic data. Without loss of generality, we generate only binary features. We collected records over multiple months using real carrier data. Each record has the number of dropped calls as well as total number of calls. We use this data to build normal distributions for the number of calls per record and

degradation probability associated with records. We then use random sampling to generate data for synthetic records from these distributions. For synthetic traces, we vary the number of features and the number of features involved in anomalies. We usually generate 10,000 records per experiment if the number of features is small. For a large number of features, we generate more records for better coverage.

Anomaly injection : For both real and synthetic data, we inject synthetic anomalies so that we can quantify the coverage (*i.e.*, fraction of anomalies that can be detected). Without loss in generality, we only insert anomalies that degrade performance. From our discussion with operations and experience with carrier data, we find that degradation across elements with same configuration/problem can vary a lot. So we use a random sampling to inject anomalies.

We inject a specified number of k-way anomalies where k varies from 1 to the total number of features. For any record that has an anomalous combination, we perturb its dropped/successful calls as follows. We compute mean (μ) and standard deviation (σ) of the dropping probability corresponding to that feature combination in the data. We derive a new dropping probability as $\mu + random(5, 10) \times \sigma$. To model sampling error, we derive the actual number of dropped calls in a record using the target dropping probability by doing random uniform sampling over the total number of calls. For example, if the target dropping probability is 10%, we randomly drop each call at 10% and count the total dropped calls.

Performance metric: We quantify the accuracy of anomaly detection using coverage. Coverage denotes the fraction of real anomalies are detected. We quantify the accuracy of root cause analysis based on the number of real root causes that we detect. We sort the probabilities of the root causes output by the NN and the top K feature combinations that have the highest probabilities are the root causes we detect. In all cases, we consider our anomaly detection and root cause analysis is correct only when the exact feature combinations match with the real ones.

V. Evaluation Results

In this section, we present evaluation results for anomaly detection and root cause analyses.

A. Real Data with Real Anomalies

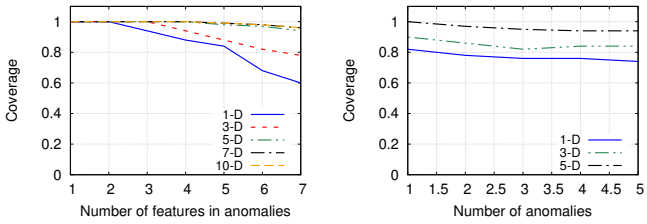
We captured several large anomalies from applying our approach to carrier data over a period of 6 months. We gave our results to the operation team and they were very interested in our findings. These anomalies were not big enough to show up at a coarse granularity. However, they affected a small subset of network with some specific features. We present a few cases in Table II where the KPI aggregated over feature combinations shows significant change but does not exhibit noticeable changes when looking at individual features or combinations of fewer features.

The first column in the table shows the features involved in the actual anomalies. For confidentiality, we omit the feature values. The anomalies that we captured have a diverse set of feature values, which means it is hard to pinpoint specific feature values that induce anomalies. The second column shows whether impact was degradation/improvement. The other columns show the maximum percentage change in drop call ratio when we look at records matching k-way combinations. For example, if anomaly is due to 3 features, 2-way column captures the maximum degradation effect when considering interaction between 2 out of 3 anomalous features. The main point is that when the performance is aggregated over fewer features than the actual number of features in the anomaly, the effect is not visible. For example, consider the first row. When aggregating at the anomalous feature combination involving 3 features, there is 813% change in the degradation probability. But the largest changes in the 1-way and 2-way combinations are only 10% and 47%, respectively. While 10% and 47% may appear high, but because the degradation probability in commercial cellular networks is very low and such increases are small in terms of the absolute value and likely to be ignored.

B. Anomaly Detection

In this section, we systematically evaluate our anomaly detection by injecting synthetic anomalies to the real and synthetic data. Synthetic anomalies are used so that we can precisely quantify the accuracy (*i.e.*, the fraction of the injected anomalies that can be accurately detected).

1) Real Data with Synthetic Anomalies: We first evaluate using the real data with synthetic anomalies injected by the method described in Section IV. We vary several parameters to understand their impact. We report the average of 50 random runs for each data point. We use coverage as our accuracy metric, which denotes the fraction of injected anomalies that we detect.



(a) Vary # features in anomalies (b) Vary # anomalies

Fig. 1. Accuracy with varying # features, anomalies and sketch dimensions.

Varying the number of features in anomalies: Intuitively, the number of features in anomalies can have significant impact on the accuracy. The anomalies involving only 1 feature (*e.g.*, $f_0 = 0$) are easier to detect than the anomalies with many features (*e.g.*, $f_0 = 0, f_1 = 0, f_2 = 0, ..., f_N = 0$). As shown in Figure 1(a), the accuracy decreases considerably with the number of features in the anomalies when the sketch dimension is 1, which corresponds to a simple pruning scheme. In comparison, by using 5 or more dimensions in sketch, we can sustain high accuracy even when the anomaly involves 7 features.

Varying number of anomalies: Next, we vary the number of anomalies from 1 to 5, where each anomaly involves up to 5 features and the total number of features is 10. As shown

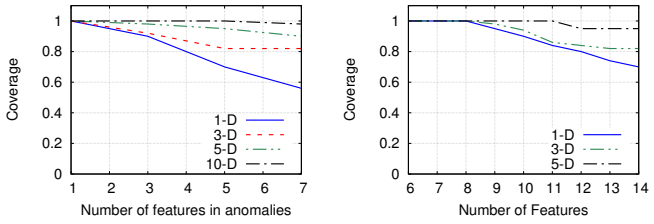
Features in Anomaly	Impact	4-way(%)	3-way(%)	2-way(%)	1-way(%)
Hardware Version, Carrier Aggregation, Morphology	Degradation		813	47	10
Hardware Version, Carrier Aggregation, RX-State	Degradation		70	15	4
Software Version, Carrier Aggregation, Morphology	Degradation		115	18	3
RX-State, Carrier Frequency	Degradation			167	2
Baseband Unit Version, Carrier Frequency, RX-State	Degradation		236	54	9
Hardware Version, Carrier Aggregation, Morphology, TX-State	Degradation	1370	148	73	4
Phone Type, Carrier Frequency	Improvement			-35	-1
Carrier Frequency, Carrier Aggregation, Morphology, TX-State	Degradation	740	118	71	3
Carrier Frequency, Carrier Aggregation, Morphology	Degradation		400	54	2
TX-State, Carrier Frequency	Degradation			140	30
Baseband Unit Version, Carrier Frequency, RX-State	Degradation		255	54	9

TABLE II
ANOMALIES FROM CARRIER DATA

in Figure 1(b), using a sketch dimension of 5 allows us to accurately detect all 5 anomalies, whereas using 1-D sketch performs considerably worse.

Running time and space: We further compare the running time and space between our lattice and complete lattice. Complete lattice contains ∼100,000 nodes and takes around 170 minutes to construct. In comparison, our lattice has 1600 nodes (45x space saving) and takes around 30 seconds (340x speedup) to build. Using 1-D sketch saves space but significantly degrades accuracy. The space required for 1-D, 3-D, and 5-D sketch are similar for 5 or fewer anomalous features, and become 188 KB, 240 KB, and 297 KB for 7 anomalous features, respectively. Storage size does not grow linearly with the sketch dimensions due to metadata storage.

2) Synthetic Data: We now evaluate using synthetic data. First, we vary the number of features in the anomalies while fixing the total number of features to 10 and the number of anomalies to 1. As shown in Figure 2(a), anomaly detection accuracy is high when sketch dimension of 5 or more is used. Reducing sketch dimension significantly degrades the accuracy. Next we vary the total number of features. Figure 2(b) shows we can detect nearly all anomalies using 5-D sketch.



(a) Vary # anomalous features (b) Vary # total features

Fig. 2. Impact of # anomalous/total features on accuracy.

C. Root Cause Analyses

Here, we evaluate our root cause diagnosis algorithm using real and synthetic data. We randomly generate 10000 samples and use 8000 of them for training. Accuracy is computed as the fraction of ground-truth anomalies that appear among top 5 feature combinations output from the diagnosis algorithms.

Unless stated otherwise, we use 8 features for real data and 10 features for synthetic data. NN uses a sketch dimension of 18 for real data and 12 for synthetic. We get these numbers empirically by finding out the minimum number of sketch dimensions required to get good accuracy. Real data requires more dimensions because features in real data take non-binary values. Root cause diagnosis requires larger sketch dimensions than anomaly detection because NN only uses sketch values from the first layer whereas anomaly detection has complete pruned lattice. Sketch dimensions of 12 and 18 are affordable since the input size grows linearly with sketch dimensions. We insert 1 anomaly involving 5 features by default. We comprehensively test our approach by varying number of anomalies, features in anomalies and sketch dimensions.

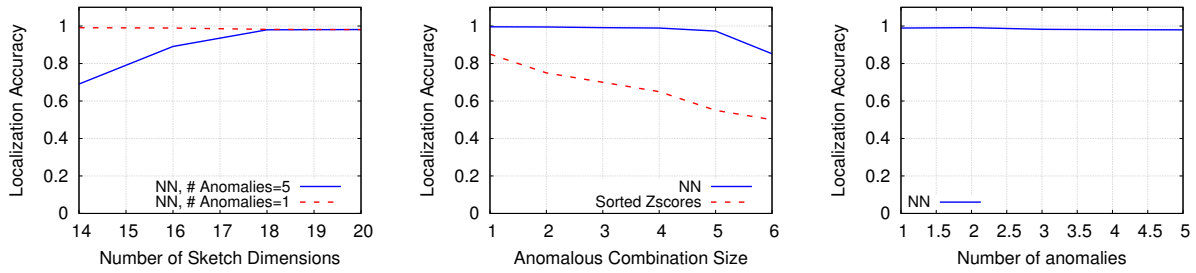
1) Real Data with Synthetic Anomalies: **Overall performance:** As shown in Figure 3, NN using 18 sketch dimensions consistently achieves close to 99% accuracy when diagnosing up to 5 anomalies, each with up to 5 features.

Impact of sketch dimension: Increasing the sketch dimension improves the accuracy of NN since it more accurately captures the performance distribution across various feature combinations. Fig. 3(a) shows the diagnosis accuracy as the sketch dimension varies from 14 to 20. For 1 anomaly, NN can diagnose 99% across all sketch dimensions. For 5 anomalies, the accuracy of NN drops to only 70% when the sketch dimension is 14, but increasing the dimension to 18 helps to improve the accuracy to 99%. This result suggests that we can select the sketch dimension according to the maximum number of features that are likely to interact.

Impact of # features in anomalies: Fig. 3(b) shows the diagnosis accuracy when the number of features in anomalies varies from 1 to 6 features in a single anomaly. The accuracy of NN is around 99% when the the number of features in anomalies is within 5 and drops to around 87% if it goes to 6. Diagnosis accuracy improves to 92% if the sketch dimension becomes 22. This shows that more features in anomalies require a higher sketch dimension for accurate diagnosis.

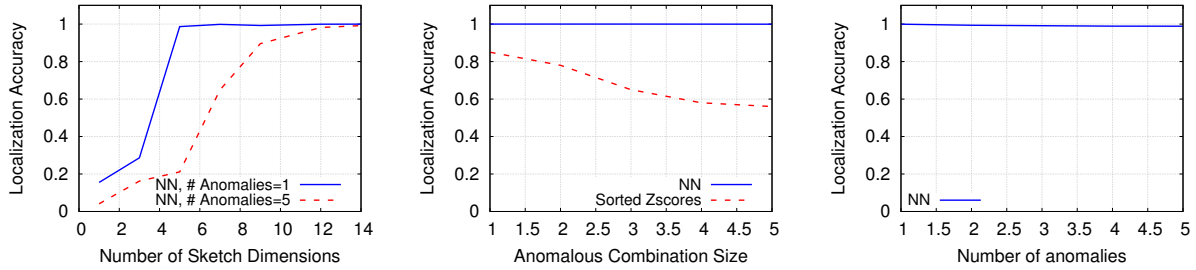
We also compare against a statistical testing scheme. For all lattice nodes, we compute zscore: $\frac{d_c - d_p}{\sqrt{d(1-d)(1/n_c + 1/n_p)}}$ where d_c/d_p is the degradation probability of nodes in the current/previous epochs, d is the combined degradation probability over all epochs, n_c/n_p is the total number of calls in current/previous epochs. This metric captures the difference in performance between the current and previous epochs. We then sort the nodes in a descending order of zscores to capture the nodes with the largest performance difference. It performs considerably worse due to its sensitivity to measurement noise and approximation errors.

Impact of the number of anomalies: In Fig. 3(c), we vary the number of anomalies from 1 to 5. By using a sketch dimension of 18, NN can achieve an accuracy close to 99%.



(a) Vary sketch dimension (b) Vary # features in anomaly (c) Vary # anomalies

Fig. 3. Root cause diagnosis accuracy for different schemes and parameters in real data.



(a) Vary sketch dimension (b) Vary # features in anomaly (c) Vary # anomalies

Fig. 4. Root cause diagnosis accuracy for different schemes and parameters in synthetic data.

2) Synthetic Data: **Overall Performance** : For synthetic data, Figure 4(b) shows that 12 sketch dimensions enable the NN to identify 99% of the root causes, each involving up to 5 features. When we increase the number of anomalies to 5, the NN can still accurately identify the root causes.

D. Result Summary

Our Lattice-Sketch can consistently detect all anomalies using 5D sketch over wide range of scenarios while exploring only $< 1\%$ of the lattice space. 1D sketch performs considerably worse due to cancellation and dilution, which make fine-grained anomalies disappear at a coarse-grained level. Our neural network based root cause diagnosis achieves close to 100% accuracy when using large enough sketch dimensions.

VI. Related Work

Anomaly detection: Statistical anomaly detection is an extensively studied topic. Various algorithms have been proposed, including time series forecast [5], smoothing, ARIMA modeling [6], and wavelet [7]. [8], [9], [10] develop probabilistic summaries to identify heavy hitters without keeping per-flow state. [2] focuses on online 2D hierarchical heavy hitter (HHH) identification where each dimension is prefix match. [11] develops both deterministic sample-based and randomized sketch-based algorithms to find HHHs using small space. [12], [13], [14] develop iterative search in lattice, but they assume that deep lying anomalies are always visible at a coarser-granularity. In HHH detection, a parent of a heavy hitter is also a heavy hitter as its traffic is the total traffic of all its children and there is no cancellation effect, which simplifies the search and pruning. In our context, a change may only be observable at a finer-granularity, but not at a coarser-granularity due to dilution and cancellation, so pruning becomes challenging. Our work is the first that addresses these concerns.

Multidimensional Anomaly Search: Research on OLAP models has developed various solutions for efficient computation of lattices[15], [16]. There are several works to make multidimensional anomaly detection scalable and efficient [17], [18], [19], [20]. They use data cubes or other structures like K-d trees to reduce dimensionality and only explore relevant subspaces in the original high-dimensional space. However, these approaches do not consider the cancellation effect that we tackle as they do not explore how performance changes across parent and child nodes. [21] presented an effective method to detect multi-dimensional heavy hitters, however it is designed for offline computation. [22], [23], [24] present algorithms to detect anomalies in multi-dimensional time series data by converting the data into multiple univariate time series. These algorithms lose information when projecting into 1D space and miss the interaction across different dimensions.

Network monitoring and diagnosis: Network performance monitoring and diagnosis has been extensively studied. Many works first build a normal profile of the service and then check divergent trends. DiffProv [25] uses differential analysis to compare anomalous and normal system behavior. [26] studies control-plane interactions in cellular networks. [27], [28], [29] use ML techniques to diagnose problems in networks but requires training on known problematic patterns. [30] develops a platform to monitor LTE radio performance using control channel. Several works study statistical dependencies between the network profile and performance [31], [32], [33], [34], [35], [36], [37]. [38] is a protocol level cellular network analytics system that deals with only UEs and base stations instead of detailed features. None of these works study multi-dimensional anomalies.

Neural networks: Neural networks have been widely used for traffic classification [39], [40], network intrusion detec-

tion [41], and video coding [42], [43]. Most existing works focus on improving the accuracy. A few recent works improve the efficiency and storage of neural networks. For example, [44], [45] propose interesting algorithms to compress the neural network structure (*e.g.*, learning only important connections, weights quantization, using Huffman coding). Our work is complimentary to these works in that we compress the input data and can be combined with the approaches that compress the neural network structure to enhance the efficiency.

VII. Conclusion

In this paper, we propose a *Lattice-Sketch* based structure to efficiently detect multi-dimensional performance impacts. We further develop a neural network based algorithm to determine the root cause for anomalies. Our extensive evaluation using real and synthetic data shows that our lattice sketch can identify nearly all impacts in our data and our sketch based Neural Network approach can diagnose root causes as long as the sketch dimension is large enough. Our technique of using Sketch to compress the input to neural network can be potentially useful beyond network diagnosis.

Acknowledgments: This work is supported in part by NSF Grant CNS-1718089 and Army Futures Command Grant W911NF1920333. We thank Prof. Sanjay Shakkottai for insightful discussions.

References

[1] C. Estan, G. Varghese, and M. Fisk, "Counting the number of active flows on a high speed link," in *Proc. of ACM SIGCOMM*, 2002.

[2] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online identification of hierarchical heavy hitters: Algorithms, evaluation, and applications," ser. IMC '04, 2004.

[3] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *PMC*, June 2017.

[4] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency momments," *J. Computer System Science*, pp. 137–147, 1999.

[5] R. S. Tsay, "Outliers, level shifts, and variance changes in time series," *Journal of Forecasting*, 1988.

[6] G. E. P. Box and G. M. Jenkins, "Time series analysis, forecasting and control," 1976,1994.

[7] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.

[8] G. Cormode and S. Muthukrishnan, "Whats hot and whats not: Tracking most frequent items dynamically," in *In Proc. ACM PODC*, 2003.

[9] ——, "Improved data stream summaries: The count-min sketch and its applications. in journal of algorithms," 2004.

[10] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *In Proc. ACM SIGCOMM*, 2002.

[11] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 4, pp. 2:1–2:48, Feb. 2008. [Online]. Available: http://doi.acm.org/10.1145/1324172.1324174

[12] R. Bhagwan, R. Kumar, and et all, "Adtributor: Revenue debugging in advertising systems," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, Seattle, WA, 2014.

[13] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu, and X. Guo, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, 2018.

[14] Q. Lin, J.-G. Lou, H. Zhang, and D. Zhang, "idice: Problem identification for emerging issues," *2016 IEEE/ACM ICSE*, 2016.

[15] K. Morfonios and Y. Ioannidis, "Cure for cubes: Cubing using a rolap engine," 01 2006, pp. 379–390.

[16] X. Li, J. Han, and H. Gonzalez, "High-dimensional olap: a minimal cubing approach," 2004.

[17] X. Li and J. Han, "Mining approximate top-k subspace anomalies in multi-dimensional time-series data," in *VLDB*, 2007.

[18] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," ser. SIGMOD '01.

[19] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD*, 2000.

[20] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: A review," *SIGKDD Explor. Newsl.*

[21] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," ser. SIGCOMM '03.

[22] M. Jones, D. Nikovski, M. Imamura, and T. Hirata, "Anomaly detection in real-valued multidimensional time series," 2014.

[23] P. Galeano, D. Peña, and R. S. Tsay, "Outlier detection in multivariate time series by projection pursuit," *Journal of the American Statistical Association*.

[24] R. Baragona and F. Battaglia, "Outliers detection in multivariate time series by independent component analysis," 2007.

[25] A. Chen, Y. Wu, A. Haeberlen, W. Zhou, and B. T. Loo, "The good, the bad, and the differences: Better network diagnostics with differential provenance," in *ACM SIGCOMM*, 2016.

[26] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu, "Control-plane protocol interactions in cellular networks," in *Proc. of SIGCOMM*, 2014.

[27] M. Z. Shafiq, L. Ji, A. Liu, J. Pang, S. Venkataraman, and J. Wang, "A first look at cellular network performance during crowded event," in *Proc. of ACM SIGMETRICS*, 2013.

[28] S. Zhou, J. Yang, and et all, "Proactive call drop avoidance in umts networks," in *INFOCOM, 2013*.

[29] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *Proceedings of the ACM SIGCOMM*, 2013.

[30] S. Kumar, E. Hamed, D. Katabi, and L. E. Li, "LTEye: LTE radio analytics made easy and accessible," in *Proc. of ACM SIGCOMM*, 2014.

[31] H. Yan, A. Flavel, and Z. G. et al., "Argus: End-to-end service anomaly detection and localization from an ISP's point of view," in *IEEE INFOCOM*, 2012.

[32] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee, "Troubleshooting chronic conditions in large IP networks," in *Proc. of ACM CoNEXT*, 2008.

[33] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with WISE," in *ACM SIGCOMM*, 2008.

[34] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao, "Towards automated performance diagnosis in a large IPTV network," in *Proc. of ACM SIGCOMM*, 2009.

[35] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *ACM SIGCOMM*, 2009.

[36] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates, "G-RCA: a generic root cause analysis platform for service quality management in large ip networks," in *ACM CoNEXT*, 2010.

[37] F. Silveira and C. Diot, "URCA: Pulling out anomalies by their root causes," in *INFOCOM*, 2010.

[38] A. Iyer, L. E. Li, and I. Stoica, "Celliq : Real-time cellular network analytics at scale," in *NSDI 15*, 2015.

[39] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on neural networks*, 2007.

[40] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[41] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the BIONETICS*, 2016.

[42] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing," *Digital Signal Processing*, vol. 72, pp. 9–18, 2018.

[43] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *OSDI 18*, 2018.

[44] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2015.

[45] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, 2017.